

# **Working Draft Project American National Standard**

**T10/1799-D**

Revision 16  
25 August 2008

---

## **Information technology - SCSI Block Commands - 3 (SBC-3)**

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (International Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.

Permission is granted to members of INCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of INCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

T10 Technical Editor:

Mark Evans  
Western Digital Corporation  
5863 Rue Ferrari  
San Jose, CA 95138  
USA

Telephone: 408-363-5257  
Email: mark.evans@wdc.com

---

Reference number  
ISO/IEC 14776-323:200x  
ANSI INCITS.\*\*\*:200x

## Points of contact

### International Committee for Information Technology Standards (INCITS) T10 Technical Committee

#### T10 Chair

John B. Lohmeyer  
LSI Logic  
4420 Arrows West Drive  
Colorado Springs, CO 80907-3444  
USA

Telephone: (719) 533-7560  
Email: lohmeyer@t10.org

T10 Web Site: <http://www.t10.org>

#### T10 Vice-Chair

Mark S. Evans  
Western Digital Corporation  
5863 Rue Ferrari  
San Jose, CA 95138  
USA

Telephone: (408) 363-5257  
Email: mark.evans@wdc.com

#### T10 E-mail reflector:

Server: majordomo@t10.org  
To subscribe send e-mail with 'subscribe' in message body  
To unsubscribe send e-mail with 'unsubscribe' in message body

#### INCITS Secretariat

Suite 200  
1250 Eye Street, NW  
Washington, DC 20005  
USA

Telephone: 202-737-8888  
Web site: <http://www.incits.org>  
Email: incits@itic.org

#### Information Technology Industry Council

Web site: <http://www.itic.org>

#### Document Distribution

INCITS Online Store  
managed by Techstreet  
1327 Jones Drive  
Ann Arbor, MI 48105  
USA

Web site: <http://www.techstreet.com/incits.html>  
Telephone: (734) 302-7801 or (800) 699-9277

Global Engineering Documents, an IHS Company  
15 Inverness Way East  
Englewood, CO 80112-5704  
USA

Web site: <http://global.ihs.com>  
Telephone: (303) 397-7956 or (303) 792-2181 or (800) 854-7179

American National Standard  
for Information Technology

## **SCSI Block Commands - 3 (SBC-3)**

Secretariat  
Information Technology Industry Council

Approved mm.dd.yy  
American National Standards Institute, Inc.

### **ABSTRACT**

This standard specifies the functional requirements for the SCSI Block Commands - 3 (SBC-3) command set. SBC-3 permits SCSI block logical units such as rigid disks to attach to computers and provides the definition for their use.

This standard maintains a high degree of compatibility with the SCSI Block Commands (SBC-2) command set, INCITS 405-2005, and while providing additional functions, is not intended to require changes to presently installed devices or existing software.

## American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

**CAUTION NOTICE:** This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

**CAUTION:** The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by

**American National Standards Institute  
11 W. 42nd Street, New York, New York 10036**

Copyright © 2004 by Information Technology Industry Council (ITI).  
All rights reserved.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission of ITI, 1250 Eye Street NW, Suite 200, Washington, DC 20005.

Printed in the United States of America

## Revision History

### R.1 Revision 0 (09 September 2005)

Revision 0 of SBC-3 is substantially equal to revision 16 of SBC-2. The only differences arise from changes made in SBC-2 discovered during the ISO process. Those changes include:

- a) Changed idle condition timer to standby condition timer in item c) of subclause 4.15.1.
- b) Changed 2 Gigabytes to 1 GiB and 2 Terabytes to 2 TiB in two places in note 10 in subclause 5.5. The 2 was change to a 1 because there are 21 bits in the LOGICAL BLOCK ADDRESS field (i.e., 1F\_FFFF is 2 097 151 that \* 512 is 1 073 741 312 which is 1 GiB).

Removed the CORRECT bit from the WRITE LONG (16) CDB as it was never supposed to be added to this command. It's not in SCSI-2 or SBC for WRITE LONG (10) and 03-383r1 did not ask for it. It showed up in sbc2r11.

### R.2 Revision 1 (16 September 2005)

- a) Incorporated the following proposals:
  - A) 04-198r5 - Background Media Scan;
  - B) 04-371r2 - SPC-4: Enable Background Operation Error Reporting Bit; and
  - C) 05-101r1 - SBC-2 Validation of Protection Information.

### R.3 Revision 2 (22 September 2005)

- a) Incorporated the following proposals:
  - A) 05-299r1 - Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2; and
  - B) 05-313r0 - SBC-3: Change to background medium scan.

### R.4 Revision 3 (16 November 2005)

- a) Incorporated the following proposals:
  - A) 05-156r7 - SBC-3, SPC-4: Application ownership of protection information Reference Tag;
  - B) 05-317r3 - SMC-3, SPC-4, SBC-3, and SSC-3: Remove Attached Media Changer model; and
  - C) 05-374r2 - SBC-3: SPC-4: Disabling Reassign on Write Long Logical Blocks.

### R.5 Revision 4 (10 February 2006)

- a) Incorporated the following proposals:
  - A) 05-157r9 - SPC Security Commands proposal (in an E-mail it was pointed out that the SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands need to be added to the SBC-3 commands list table);
  - B) 05-340r3 - SBC-3 SPC-4 Background scan additions;
  - C) 05-368r2 - SPC-4 SBC-3 SMC-3 Allow more commands through Write Exclusive reservations; and
  - D) 05-383r4 - SPC-4: Deferred microcode downloads.

In addition the following editorial corrections were received from E-mail were incorporated:

- a) the initiator control (IC) enable bit description had the SIZE bit polarity backwards;
- b) changed filed to field;
- c) WRITE SAME (32) was placed into the list of medium access commands in the Protection types overview subclause;
- d) the field name P\_TYPEABLE in table 19 (FMTPINFO bit, RTO\_REQ bit, and PROTECTION FIELD USAGE field) footnote d was changed to P\_TYPE; and
- e) the field name DATA BLOCK GUARD in table 80 (LBDATA bit and PBDATA bit) in row 0 0 was changed to LOGICAL BLOCK GUARD.

### R.6 Revision 5 (11 May 2006)

- a) Incorporated the following proposals:

- A) 06-248r1 - Proposal to remove the PREVENT ALLOW MEDIUM REMOVAL (PAMR) command from SPC-4

In addition the following editorial corrections were received from E-mail were incorporated:

- a) The following sentence occurs on SBC-3 rev. 3, page 122, first paragraph under table 110, last sentence of paragraph; and also on page 124, first paragraph under table 111, last sentence of paragraph:

“To determine the number of blocks at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.”

In both cases, the “MODE SELECT” needs to be replaced with “MODE SENSE”. To use the “select” operation is nonsensical if the purpose is to discover the current block size of the disk drive.

## R.7 Revision 6 (24 July 2006)

- a) Incorporated the following proposals:
  - A) 06-259r1 - SAM-4, et al.: making linked commands obsolete;
  - B) 06-274r1 - SPC-4 SBC-3 REQUEST SENSE and Stopped power condition; and
  - C) 06-323r1 - SAM-4 SPC-4 et al Multiple service delivery subsystem editorial tweaks.

## R.8 Revision 7 (22 September 2006)

- a) Incorporated the following proposals:
  - A) 06-034r5 - SBC-3 Physical blocks; and
  - B) 06-350r0 - SPC-4/SBC-3: Power conditions state machine clarification.

In addition the following editorial corrections were received from E-mail were incorporated:

- a) In the Background Scan Results log page (section 6.2.2) in table 101 on page 119, there seems to be a missing “scan” in the row for code 08h. It currently reads:

“Background medium halted, waiting...”.

GAH: Agreed, it should read “Background medium scan halted, waiting...”. I note that the word “medium” is not present in the descriptions of other code rows, maybe that word should be removed for consistency.

GOP: The word “medium” was added to all the descriptions of other code rows to make it consistent with the rest of the clause.

- b) In “Background medium scan parameter format” for parameter code 0001h to 0800h shown in table 102 the “accumulated power on minutes” field is defined on page 121 as “The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing.” That is the same definition of the same named field found in the “Background medium scan status parameter format” (for parameter code 0) shown in table 99. Shouldn't the description associated with table 102 be referencing when the error associated with that parameter number was logged [similar to the way the self test log page outputs its results]?

GAH: I agree. It should say “The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing at the time the background scan error occurred.”

## R.9 Revision 8 (18 January 2007)

Fixed the byte numbering in the READ CAPACITY (16) parameter data table in 5.13.2.

Based on the following E-mail from Mark Evans pointing out error in the incorporation of proposal 05-340r3 changes were made to table 107 and table 110 and a description of the DS bit and the SPF bit were placed under table 104:

While going through SBC-3, I see that table 102 (parameter control bits for Background Scanning Status log parameter) looks goofy. I think it should look like table 194 in SAS. I'm making that change in our internal document and recommend that you make the change in SBC.

Incorporated the following proposals:

- a) 06-479r1 - Mandate CAPACITY DATA HAS CHANGED unit attention; and
- b) 06-393r3 - On-disk bitmap support.

## R.10 Revision 8a (18 January 2007)

The ORWRITE CDB table title was fixed and the ORWRITE operation code in the CDB was corrected.

## R.11 Revision 9 (22 March 2007)

Based on the following 1/23/2007 E-mail from Mark Evans pointing out a duplicate entry between subclause 4.17 Protection information model and subclause 5.31 WRITE AND VERIFY (10) command. The duplicate wording that needs to be deleted is from 5.31 WRITE AND VERIFY (10) command and is:

If the P\_TYPE bit is set to one in the READ CAPACITY (16) parameter data (see 5.13), the device server shall terminate this command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE. During the investigation of this more duplicate wording was found in subclause 5.38 WRITE SAME (16) command. This duplication wording that needs to be deleted is:

If the P\_TYPE bit is set to one in the READ CAPACITY (16) parameter data (see 5.13), the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE

In both cases the duplicate wording should have been removed as part of T10/05-156 revision 7.

Based on the following 1/26/2007 E-mail from Rob Elliott editorial changes were made as recommended in the e-mail. The only recommend change not made was to change ORWRITE to ORWRITE (16). Instead all the ORWRITE (16)s were changed to ORWRITE.

- 1) On page 55, this text needs small caps and the table references need to be fixed:  
The device server shall:
  - a) check protection information read from the medium based on the ORPROTECT field as described in tableyy2; and
  - b) check protection information transferred from the data-out buffer based on the ORPROTECT field as described in table yy3.
- 2) In table 3 (reservations), on page 29, and on page 30, ORWRITE s/b ORWRITE (16)
- 3) In the table of tables on page xiii, the prevent field doesn't show up in small caps like the others.
- 4) On page 34 and 35, several of the equations are truncated on the left and right. The reason is the font size grew - the Equation format or character format is messed up on those lines.

Based on the following 2/8/2007 emails from Dave Peterson and Rob Elliott pointing out a badly worded sentence that sentence was corrected.

Dave Peterson:

Sentence in question:

Subclause 5.2.2.3 first paragraph, first sentence on physical page 52:

"When the SI bit is set to one, the device server need not write the initialization pattern over the header and other header and other parts of the medium not previously accessible to the application client."

The extra "...and other header..." appears odd. Please clarify.

Rob Elliott:

I agree with David's fix.

Based on the following email from Doug Gilbert received on 2/23/2007 I added in cross-references to the footnote in table 84. The cross-references were added to all the yes terms in the 3rd column.

George, you should delete the spurious "and other header" words from the SI bit paragraph SBC-3. That was broken in sbc2r12 while incorporating editor's meeting comments to fix a parenthetical expression without an e.g. or i.e.

In section 5.35 on WRITE LONG(10) there is table 84 which has a footnote "a". It is hanging because there seems nothing in the main table it refers to. My guess is that it refers to the third column header which starts "More than one logical block...".

Incorporated the following proposals:

- a) 07-110r0 - Update Block Limits VPD Page for ORWRITE; and
- b) 07-113r0 - Maximum transfer sizes for XPWRITE XDWRITE XDREAD PRE-FETCH.

## R.12 Revision 10 (17 May 2007)

Incorporated the following proposals:

- a) 07-203r0 - SBC-3 SPC-4 Block Device Characteristics VPD page and medium rotation rate field; and
- b) 07-208r0 - SBC-3 Rename field in READ CAPACITY(16) parameter data.

## R.13 Revision 11 (19 July 2007)

All mode page format tables have been updated to include the SubPage Format (SPF) bit.

Incorporated the following proposals:

- a) 07-257r1 - Prohibited needed as a keyword in SPC-4;
- b) 07-271r1 - SBC-3, Clarifications for Background Scan Results log page;
- c) 07-281r1 - SPC-4, SBC-3, #Except INQUIRY, REPORT LUNS, and REQUEST SENSE#; and
- d) 07-302r1 - SBC-3 WRITE LONG Additional Sense code option to support SAT-2.

Based on the following email from Rob Elliott received on 5/16//2007:

1. SBC-3 uses both "media access commands" (from changes made in the DIF area) and "medium access commands." SBC-2 only used "medium access commands."

I changed all the "media access commands" to "medium access commands".

As a result of 07-271 references to generic fields (e.g., OPERATION CODE field, PAGE CODE field, PAGE LENGTH field) were added under the tables where those fields were defined.

## R.14 Revision 12 (11 November 2007)

Incorporated the following proposals:

- a) 07-472r0 - Reporting nominal form factor;
- b) 07-447r1 - Read-Write Error Recovery clarifications; and
- c) 07-481r1 - Mention that DIF equals protection information.

Made editorial changes based on the following email received from Rob Elliott on 7/24/2007

You added wording like this to many sections:

"The OPERATION CODE field is defined in SPC-4 shall be set to the value defined in table 72." but "shall" needs to be "and shall".

I see a few that are broken, though: After table 81, it points to table 72. After table 91, it points to table 90.

The wording for SPF bit in the mode pages:

"A SubPage Format (SPF) bit set to zero indicates that the page\_0 mode page format is being used (see SPC-4)." should probably be:

"The SubPage Format (SPF) bit is defined in SPC-4 and shall be set to the value defined in table xx."xx."" (probably combined with the PAGE CODE sentence in all cases)

Table 102, bytes 4-19 - parameters s/b parameter (there's only one)

Table 104 - add (MSB)/(LSB) on the last field

Table 112 - in the sentence after this table, the period is on the wrong line.

Table 113, 115 - change 3h to 03h

Editor's note 1: Either that proposal or the editor deleted the wrong line from SBC-2 when obsoleting



linked commands. It should just be "CONDITION MET" not "INTERMEDIATE-CONDITION MET". See SBC-2's original wording - the LINK bit set to zero sentences are the ones that should have remained. Added the control byte reference to SAM-4 for all the CDBs.

Made editorial changes as noted in the following email received from Gerry Houlder on 11/06/2007:

This inconsistent behavior was noted by an engineer at Seagate while reviewing Protection Information behavior in SBC-3. It would seem to be more appropriate to call out 05/24 sense instead of 05/20 to be consistent with products that report 05/24 for reserved fields.

### **R.15 Revision 13 (24 January 2008)**

Made all binary and hexadecimal numbers have a consistent format, separating groups of four digits with underscores. Changed the conventions clause (see clause 3.4) to define this format.

Incorporated the following proposals:

- a) 07-451r1 - WRITE LONG COR\_DIS and WR\_UNCOR interaction;
- b) 07-454r5 - Capability based Command Security;
- c) 07-459r4 - Unit attention condition queuing;
- d) 08-041r1 - Use period as decimal separator in T10 standards;

Made editorial changes as noted in the following email received from Rob Elliott on 01/03/2008:

- add the subpage code column to the table of log page codes
- add 00h/FFh as Supported Log Pages and Subpages to the table of log page codes
- add 01h-3Eh/FFh as Supported Subpages to the table of log page codes
- add 18h/00h-3Eh as Protocol Specific Port log pages to the table of log page codes
- add the DS bit and SPF bit (0b) to byte 0 of any log page definitions
- add the SUBPAGE CODE field (00h) to byte 1 of any log page definitions
- make sure the command set table agrees with the SPC-4 annex about the page names, reserved and vendor-specific ranges, etc.

### **R.16 Revision 14 (20 March 2008)**

Incorporated the following proposals:

- a) 08-139r1 - START STOP UNIT command additions.

Made the editorial change in 4.16 based on an email from Fred Knight: changed "idle" to "standby" in list item (C) in the bulleted list for how to process a REQUEST SENSE command while in the standby power condition.

Updated the definition of the PREVENT ALLOW MEDIUM REMOVAL command to be as proposed in 05-317r3 based on emails from Rob Elliott.

Made other editorial changes as denoted by change bars, including:

- added definitions for device server, I\_T nexus, logical unit, SCSI target device, and unit attention condition
- added requirement that, if the medium in a direct-access block device is removable, and the medium is removed, then the device server shall establish a unit attention condition

Made other minor editorial and formatting changes not indicated by change bars.

### **R.17 Revision 15 (15 May 2008)**

Incorporated the following proposals:

- a) 08-156r2 -Non-volatile cache becoming volatile.

Made other minor editorial and formatting changes not indicated by change bars.

## R.18 Revision 16 (25 August 2008)

Incorporated the following proposals:

- a) 08-116r3 - SBC-3 SPC-4: Protection Type 3 Reference Tag Clarification

Based on an email exchange between Mark Evans and Rob Elliott, changed the following sentence in 4.9 Medium defects from, "During a self-test operation (see SPC-4), the device server shall ignore pseudo unrecoverable errors with correction disabled and shall process the pseudo unrecoverable errors with correction disabled." to "During a self-test operation (see SPC-4), the device server shall ignore pseudo unrecoverable errors with correction disabled and shall process the pseudo unrecoverable errors with correction enabled."

The conventions subclause was updated to include the latest descriptions of the conventions for lettered and numbered lists.

The definition for how a range of numeric values are represented in this standard was added, and the attempt was made to find all instances of "through", "-", and other representations where they were used to represent a range of numbers and replaced each instance with "to" to conform to this definition.

Other minor editorial and formatting changes were made that are not indicated by change bars.

## Contents

	Page
1 Scope .....	1
2 Normative References .....	2
2.1 Normative references overview .....	2
2.2 Approved references .....	2
2.3 References under development .....	3
3 Definitions, symbols, abbreviations, keywords, and conventions .....	4
3.1 Definitions .....	4
3.2 Symbols and abbreviations .....	7
3.3 Keywords .....	7
3.4 Conventions .....	8
3.4.1 Editorial conventions .....	8
3.4.2 Numeric conventions .....	9
3.4.3 Lists conventions .....	9
3.4.3.1 Lists conventions overview .....	9
3.4.3.2 Unordered lists .....	10
3.4.3.3 Ordered lists .....	10
3.4.4 Precedence .....	10
4 Direct-access block device type model .....	11
4.1 Direct-access block device type model overview .....	11
4.2 Media examples .....	11
4.2.1 Media examples overview .....	11
4.2.2 Rotating media .....	11
4.2.3 Memory media .....	12
4.3 Removable medium .....	12
4.3.1 Removable medium overview .....	12
4.3.2 Removable medium with an attached media changer .....	13
4.4 Logical blocks .....	13
4.5 Physical blocks .....	14
4.6 Ready state .....	16
4.7 Initialization .....	17
4.8 Write protection .....	17
4.9 Medium defects .....	18
4.10 Write failures .....	19
4.11 Caches .....	19
4.12 Implicit HEAD OF QUEUE command processing .....	21
4.13 Reservations .....	21
4.14 Error reporting .....	23
4.14.1 Error reporting overview .....	23
4.14.2 Processing pseudo unrecovered errors .....	24
4.14.3 Block commands sense data descriptor .....	24
4.15 Model for XOR commands .....	25
4.15.1 Model for XOR commands overview .....	25
4.15.2 Storage array controller supervised XOR operations .....	25
4.15.2.1 Storage array controller supervised XOR operations overview .....	25
4.15.2.2 Update write operation .....	25
4.15.2.3 Regenerate operation .....	26
4.15.2.4 Rebuild operation .....	26
4.15.3 Array subsystem considerations .....	27
4.15.3.1 Array subsystem considerations overview .....	27
4.15.3.2 Buffer full status handling .....	27
4.15.3.3 Access to an inconsistent stripe .....	27

4.15.4 XOR data retention requirements .....	27
4.16 START STOP UNIT and power conditions.....	28
4.16.1 START STOP UNIT and power conditions overview.....	28
4.16.2 START STOP UNIT and power conditions state machine.....	29
4.16.2.1 START STOP UNIT and power conditions state machine overview .....	29
4.16.2.2 SSU_PC0:Powered_on state .....	30
4.16.2.2.1 SSU_PC0:Powered_on state description .....	30
4.16.2.2.2 Transition SSU_PC0:Powered_on to SSU_PC1:Active .....	30
4.16.2.2.3 Transition SSU_PC0:Powered_on to SSU_PC4:Stopped.....	30
4.16.2.3 SSU_PC1:Active state .....	31
4.16.2.3.1 SSU_PC1:Active state description .....	31
4.16.2.3.2 Transition SSU_PC1:Active to SSU_PC2:Idle.....	31
4.16.2.3.3 Transition SSU_PC1:Active to SSU_PC3:Standby .....	31
4.16.2.3.4 Transition SSU_PC1:Active to SSU_PC4:Stopped .....	31
4.16.2.4 SSU_PC2:Idle state .....	31
4.16.2.4.1 SSU_PC2:Idle state description .....	31
4.16.2.4.2 Transition SSU_PC2:Idle to SSU_PC1:Active .....	31
4.16.2.4.3 Transition SSU_PC2:Idle to SSU_PC3:Standby .....	32
4.16.2.4.4 Transition SSU_PC2:Idle to SSU_PC4:Stopped .....	32
4.16.2.5 SSU_PC3:Standby state .....	32
4.16.2.5.1 SSU_PC3:Standby state description .....	32
4.16.2.5.2 Transition SSU_PC3:Standby to SSU_PC1:Active .....	32
4.16.2.5.3 Transition SSU_PC3:Standby to SSU_PC2:Idle .....	32
4.16.2.5.4 Transition SSU_PC3:Standby to SSU_PC4:Stopped.....	32
4.16.2.6 SSU_PC4:Stopped state.....	32
4.16.2.6.1 SSU_PC4:Stopped state description.....	32
4.16.2.6.2 Transition SSU_PC4:Stopped to SSU_PC1:Active .....	33
4.16.2.6.3 Transition SSU_PC4:Stopped to SSU_PC2:Idle .....	33
4.16.2.6.4 Transition SSU_PC4:Stopped to SSU_PC3:Standby.....	33
4.17 Protection information model.....	33
4.17.1 Protection information overview.....	33
4.17.2 Protection types .....	33
4.17.2.1 Protection types overview .....	33
4.17.2.2 Type 0 protection.....	34
4.17.2.3 Type 1 protection.....	35
4.17.2.4 Type 2 protection.....	35
4.17.2.5 Type 3 protection.....	36
4.17.3 Protection information format.....	36
4.17.4 Logical block guard .....	38
4.17.4.1 Logical block guard overview .....	38
4.17.4.2 CRC generation.....	38
4.17.4.3 CRC checking .....	39
4.17.4.4 CRC test cases .....	39
4.17.5 Application of protection information.....	39
4.17.6 Protection information and commands .....	40
4.18 Grouping function .....	40
4.19 Background scanning operations .....	40
4.19.1 Background scanning overview .....	40
4.19.2 Background pre-scan.....	41
4.19.2.1 Enabling the background pre-scan operation.....	41
4.19.2.2 Suspending the background pre-scan operation.....	41
4.19.2.3 Halting the background pre-scan operation .....	41
4.19.3 Background medium scan .....	42
4.19.3.1 Enabling the background medium scan operation .....	42
4.19.3.2 Suspending the background medium scan operation .....	42
4.19.3.3 Halting the background medium scan operation .....	42
4.19.4 Interpreting the logged results .....	43

4.20 Association between commands and CbCS permission bits .....	43
4.21 Deferred microcode activation .....	45
5 Commands for direct-access block devices .....	46
5.1 Commands for direct-access block devices overview .....	46
5.2 FORMAT UNIT command .....	49
5.2.1 FORMAT UNIT command overview .....	49
5.2.2 FORMAT UNIT parameter list .....	52
5.2.2.1 FORMAT UNIT parameter list overview .....	52
5.2.2.2 Parameter list header .....	52
5.2.2.3 Initialization pattern descriptor .....	56
5.2.2.4 Address descriptor formats .....	57
5.2.2.4.1 Address descriptor formats overview .....	57
5.2.2.4.2 Short block format address descriptor .....	58
5.2.2.4.3 Long block format address descriptor .....	58
5.2.2.4.4 Bytes from index format address descriptor .....	59
5.2.2.4.5 Physical sector format address descriptor .....	59
5.3 ORWRITE command .....	60
5.4 PRE-FETCH (10) command .....	65
5.5 PRE-FETCH (16) command .....	67
5.6 PREVENT ALLOW MEDIUM REMOVAL command .....	67
5.7 READ (6) command .....	68
5.8 READ (10) command .....	71
5.9 READ (12) command .....	75
5.10 READ (16) command .....	75
5.11 READ (32) command .....	76
5.12 READ CAPACITY (10) command .....	77
5.12.1 READ CAPACITY (10) overview .....	77
5.12.2 READ CAPACITY (10) parameter data .....	78
5.13 READ CAPACITY (16) command .....	79
5.13.1 READ CAPACITY (16) command overview .....	79
5.13.2 READ CAPACITY (16) parameter data .....	80
5.14 READ DEFECT DATA (10) command .....	81
5.14.1 READ DEFECT DATA (10) command overview .....	81
5.14.2 READ DEFECT DATA (10) parameter data .....	82
5.15 READ DEFECT DATA (12) command .....	83
5.15.1 READ DEFECT DATA (12) command overview .....	83
5.15.2 READ DEFECT DATA (12) parameter data .....	83
5.16 READ LONG (10) command .....	84
5.17 READ LONG (16) command .....	85
5.18 REASSIGN BLOCKS command .....	86
5.18.1 REASSIGN BLOCKS command overview .....	86
5.18.2 REASSIGN BLOCKS parameter list .....	87
5.19 START STOP UNIT command .....	89
5.20 SYNCHRONIZE CACHE (10) command .....	91
5.21 SYNCHRONIZE CACHE (16) command .....	92
5.22 VERIFY (10) command .....	93
5.23 VERIFY (12) command .....	103
5.24 VERIFY (16) command .....	103
5.25 VERIFY (32) command .....	104
5.26 WRITE (6) command .....	105
5.27 WRITE (10) command .....	106
5.28 WRITE (12) command .....	110
5.29 WRITE (16) command .....	110
5.30 WRITE (32) command .....	111
5.31 WRITE AND VERIFY (10) command .....	112
5.32 WRITE AND VERIFY (12) command .....	113

5.33 WRITE AND VERIFY (16) command .....	113
5.34 WRITE AND VERIFY (32) command .....	114
5.35 WRITE LONG (10) command.....	115
5.36 WRITE LONG (16) command.....	118
5.37 WRITE SAME (10) command.....	119
5.38 WRITE SAME (16) command.....	121
5.39 WRITE SAME (32) command.....	122
5.40 XDREAD (10) command .....	123
5.41 XDREAD (32) command .....	124
5.42 XDWRITE (10) command.....	125
5.43 XDWRITE (32) command.....	126
5.44 XDWRITEREAD (10) command.....	127
5.45 XDWRITEREAD (32) command.....	128
5.46 XPWRITE (10) command.....	129
5.47 XPWRITE (32) command.....	130
6 Parameters for direct-access block devices.....	131
6.1 Diagnostic parameters.....	131
6.1.1 Diagnostic parameters overview .....	131
6.1.2 Translate Address Output diagnostic page.....	131
6.1.3 Translate Address Input diagnostic page.....	132
6.2 Log parameters .....	134
6.2.1 Log parameters overview.....	134
6.2.2 Background Scan Results log page.....	135
6.2.3 Format Status log page.....	141
6.2.4 Non-volatile Cache log page.....	142
6.3 Mode parameters .....	144
6.3.1 Mode parameters overview.....	144
6.3.2 Mode parameter block descriptors.....	146
6.3.2.1 Mode parameter block descriptors overview .....	146
6.3.2.2 Short LBA mode parameter block descriptor .....	146
6.3.2.3 Long LBA mode parameter block descriptor .....	148
6.3.3 Background Control mode page .....	149
6.3.4 Caching mode page.....	151
6.3.5 Read-Write Error Recovery mode page.....	155
6.3.6 Verify Error Recovery mode page.....	161
6.3.7 XOR Control mode page.....	162
6.4 Vital product data (VPD) parameters.....	163
6.4.1 VPD parameters overview .....	163
6.4.2 Block Limits VPD page .....	163
6.4.3 Block Device Characteristics VPD page .....	164
Annex A (informative) Numeric order codes .....	166
A.1 Variable length CDBs.....	166
A.2 Service action CDBs .....	166
Annex B (informative) XOR command examples.....	168
B.1 XOR command examples overview .....	168
B.2 Update write operation .....	168
B.3 Regenerate operation .....	169
B.4 Rebuild operation .....	170
Annex C (informative) CRC example in C.....	172

## Tables

	Page
1 Standards bodies .....	2
2 Numbering convention examples .....	9
3 SBC-2 commands that are allowed in the presence of various reservations .....	22
4 Example error conditions .....	23
5 Sense data field usage for direct-access block devices .....	23
6 Block commands sense data descriptor format .....	24
7 User data and protection information format .....	36
8 Contents of the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer .....	37
9 Setting the LOGICAL BLOCK REFERENCE TAG field of the subsequent logical blocks in the data-in buffer and/or data-out buffer .....	37
10 CRC polynomials .....	38
11 CRC test cases .....	39
12 Associations between commands and CbCS permissions .....	43
13 Commands for direct-access block devices .....	46
14 Obsolete commands for direct-access block devices .....	48
15 FORMAT UNIT command .....	49
16 FORMAT UNIT command address descriptor usage .....	51
17 FORMAT UNIT parameter list .....	52
18 Short parameter list header .....	52
19 Long parameter list header .....	53
20 FMTPINFO field and PROTECTION FIELD USAGE field .....	53
21 Initialization pattern descriptor .....	56
22 Initialization pattern modifier (IP MODIFIER) field .....	56
23 INITIALIZATION PATTERN TYPE field .....	57
24 Address descriptor formats .....	58
25 Short block format address descriptor (000b) .....	58
26 Long block format address descriptor (011b) .....	58
27 Bytes from index format address descriptor (100b) .....	59
28 Physical sector format address descriptor (101b) .....	59
29 ORWRITE command .....	60
30 ORPROTECT field - checking protection information read from the medium .....	61
31 ORPROTECT field - checking protection information from the data-out buffer .....	63
32 PRE-FETCH (10) command .....	65
33 PRE-FETCH (16) command .....	67
34 PREVENT ALLOW MEDIUM REMOVAL command .....	67
35 PREVENT field .....	68
36 READ (6) command .....	68
37 Protection information checking for READ (6) .....	70
38 READ (10) command .....	71
39 RDPROTECT field .....	71
40 Force unit access for read operations .....	74
41 READ (12) command .....	75
42 READ (16) command .....	75
43 READ (32) command .....	76
44 READ CAPACITY (10) command .....	77
45 READ CAPACITY (10) parameter data .....	78
46 READ CAPACITY (16) command .....	79
47 READ CAPACITY (16) parameter data .....	80
48 P_TYPE field and PROT_EN bit .....	80
49 LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field .....	80
50 READ DEFECT DATA (10) command .....	81
51 READ DEFECT DATA (10) parameter data .....	82
52 READ DEFECT DATA (12) command .....	83
53 READ DEFECT DATA (12) parameter data .....	83

54 READ LONG (10) command .....	84
55 READ LONG (16) command .....	86
56 REASSIGN BLOCKS command .....	87
57 REASSIGN BLOCKS parameter list .....	87
58 REASSIGN BLOCKS short parameter list header .....	87
59 REASSIGN BLOCKS long parameter list header .....	88
60 START STOP UNIT command .....	89
61 POWER CONDITION MODIFIER field .....	89
62 POWER CONDITION field .....	90
63 SYNCHRONIZE CACHE (10) command .....	91
64 SYNC_NV bit .....	92
65 SYNCHRONIZE CACHE (16) command .....	92
66 VERIFY (10) command .....	93
67 VRPROTECT field with BYCHK set to zero - checking protection information read from the medium .....	94
68 VRPROTECT field with BYCHK set to one - checking protection information read from the medium .....	97
69 VRPROTECT field with BYCHK set to one - checking protection information from the data-out buffer .....	99
70 VRPROTECT field with BYCHK set to one - byte-by-byte comparison requirements .....	101
71 VERIFY (12) command .....	103
72 VERIFY (16) command .....	103
73 VERIFY (32) command .....	104
74 WRITE (6) command .....	105
75 WRITE (10) command .....	106
76 WRPROTECT field .....	106
77 Force unit access for write operations .....	109
78 WRITE (12) command .....	110
79 WRITE (16) command .....	110
80 WRITE (32) command .....	111
81 WRITE AND VERIFY (10) command .....	112
82 WRITE AND VERIFY (12) command .....	113
83 WRITE AND VERIFY (16) command .....	113
84 WRITE AND VERIFY (32) command .....	114
85 WRITE LONG (10) command .....	115
86 COR_DIS bit, WR_UNCOR bit, and PBLOCK bit .....	116
87 WRITE LONG (16) command .....	118
88 WRITE SAME (10) command .....	119
89 LBDATA bit and PBDATA bit .....	120
90 WRITE SAME (16) command .....	121
91 WRITE SAME (32) command .....	122
92 XDREAD (10) command .....	123
93 XDREAD (32) command .....	124
94 XDWRITE (10) command .....	125
95 XDWRITE (32) command .....	126
96 XDWRITEREAD (10) command .....	127
97 XDWRITEREAD (32) command .....	128
98 XPWRITE (10) command .....	129
99 XPWRITE (32) command .....	130
100 Diagnostic page codes .....	131
101 Translate Address Output diagnostic page .....	131
102 Translate Address Input diagnostic page .....	132
103 Log page codes .....	134
104 Background Scan Results log page .....	135
105 Background Scan Results log page parameter codes .....	135
106 Background Scanning Status parameter format .....	136
107 Parameter control byte for Background Scanning Status log parameter .....	137
108 BACKGROUND SCANNING STATUS field .....	137
109 Background Medium Scan parameter format .....	138
110 Parameter control byte for Background Medium Scan log parameter .....	139



111 REASSIGN STATUS field .....	140
112 Format Status log page parameter codes .....	141
113 Non-volatile Cache log page .....	142
114 Non-volatile Cache log parameters .....	143
115 Remaining Non-volatile Time parameter data .....	143
116 REMAINING NON-VOLATILE TIME field .....	143
117 Maximum Non-volatile Time parameter data .....	143
118 MAXIMUM NON-VOLATILE TIME field .....	144
119 DEVICE-SPECIFIC PARAMETER field for direct-access block devices .....	144
120 Mode page codes for direct-access block devices .....	145
121 Short LBA mode parameter block descriptor .....	146
122 Long LBA mode parameter block descriptor .....	148
123 Background Control mode page .....	149
124 Caching mode page .....	151
125 DEMAND READ RETENTION PRIORITY field .....	152
126 WRITE RETENTION PRIORITY field .....	153
127 Read-Write Error Recovery mode page .....	155
128 Combined error recovery bit descriptions .....	157
129 Verify Error Recovery mode page .....	161
130 XOR Control mode page .....	162
131 Direct-access block device VPD page codes .....	163
132 Block Limits VPD page .....	163
133 Block Device Characteristics VPD page .....	164
134 MEDIUM ROTATION RATE field .....	165
135 NOMINAL FORM FACTOR field .....	165
A.1 Variable length command service action code assignments .....	166
A.2 SERVICE ACTION IN (16) service actions .....	166
A.3 SERVICE ACTION OUT (16) service actions .....	167

## Figures

	Page
1 SCSI document relationships .....	1
2 Logical blocks and physical blocks examples .....	15
3 Logical block to physical block alignment examples .....	16
4 Power condition state machine for logical units implementing the START STOP UNIT command .....	30
B.1 Update write operation (storage array controller supervised) .....	169
B.2 Regenerate operation (storage array controller supervised) .....	170
B.3 Rebuild operation (storage array controller supervised) .....	171

**Foreword (This foreword is not part of this standard)**

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, International Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the International Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, INCITS had the following members:

Karen Higginbottom, Chair

David Michael, Vice-Chair

INCITS Technical Committee T10 - SCSI Storage Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair

Mark S. Evans, Vice-Chair

Ralph O. Weber, Secretary

## Introduction

The standard is organized as follows:

- Clause 1 (Scope) describes the relationship of this standard to the SCSI family of standards.
- Clause 2 (Normative References) provides references to other standards and documents.
- Clause 3 (Definitions, symbols, abbreviations, keywords, and conventions) defines terms and conventions used throughout this standard.
- Clause 4 (Direct-access block device type model) provides an overview of the direct-access block device type and the command set.
- Clause 5 (Commands for direct-access block devices) defines commands specific to direct-access block devices.
- Clause 6 (Parameters for direct-access block devices) defines diagnostic pages, mode parameters and pages, log pages, and VPD pages specific to direct-access block devices.

Informative Annex A (Numeric order codes) summarizes service action assignments for variable-length commands and commands using the SERVICE ACTION IN and SERVICE ACTION OUT operation codes.

Informative Annex B (XOR command examples) provides examples of XOR command usage.

Informative Annex C (CRC example in C) provides example C code for the protection information CRC.

## American National Standard for Information Technology -

# SCSI Block Commands - 3 (SBC-3)

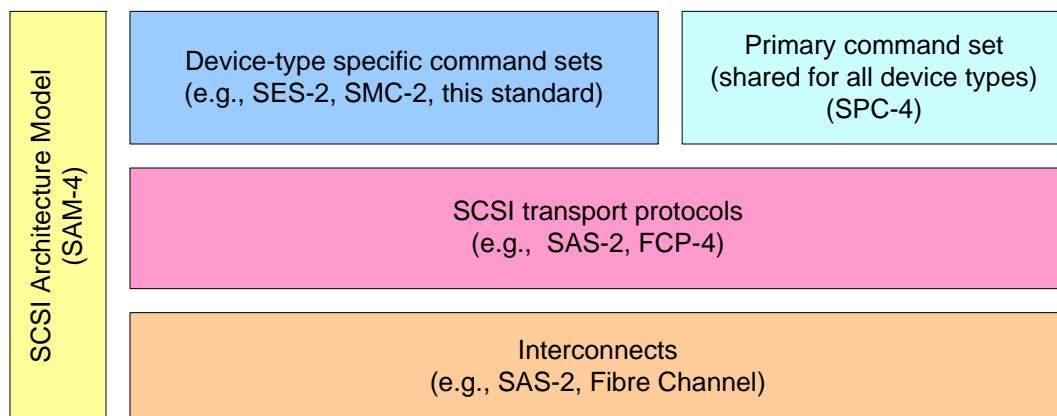
## 1 Scope

This standard defines the command set extensions to facilitate operation of SCSI direct-access block devices. The clauses of this standard, implemented in conjunction with the applicable clauses of SPC-4, fully specify the standard command set for SCSI direct-access block devices.

The objective of this standard is to:

- a) permit an application client to communicate over a SCSI service delivery subsystem with a logical unit that declares itself to be a direct-access block device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-4); and
- b) define commands unique to the direct-access block device type.

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards.



**Figure 1 — SCSI document relationships**

Figure 1 is intended to show the general relationship of the documents to one another, and is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

This standard makes obsolete the following concepts from previous standards:

- a) linked commands.

## 2 Normative References

### 2.1 Normative references overview

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI:

- a) approved ANSI standards;
- b) approved and draft international and regional standards (e.g., ISO, IEC, CEN/CENELEC, ITU-T); and
- c) approved and draft foreign standards (e.g., BSI, JIS, and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at <http://www.ansi.org>.

Additional availability contact information is provided below as needed.

Table 1 lists standards bodies and their web sites.

**Table 1 — Standards bodies**

Abbreviation	Standards body	Web site
ANSI	American National Standards Institute	<a href="http://www.ansi.org">http://www.ansi.org</a>
BSI	British Standards Institution	<a href="http://www.bsi-global.com">http://www.bsi-global.com</a>
CEN	European Committee for Standardization	<a href="http://www.cenorm.be">http://www.cenorm.be</a>
CENELEC	European Committee for Electrotechnical Standardization	<a href="http://www.cenelec.org">http://www.cenelec.org</a>
DIN	German Institute for Standardization	<a href="http://www.din.de">http://www.din.de</a>
IEC	International Engineering Consortium	<a href="http://www.iec.ch">http://www.iec.ch</a>
IEEE	Institute of Electrical and Electronics Engineers	<a href="http://www.ieee.org">http://www.ieee.org</a>
INCITS	International Committee for Information Technology Standards	<a href="http://www.incits.org">http://www.incits.org</a>
ISO	International Standards Organization	<a href="http://www.iso.ch">http://www.iso.ch</a>
ITI	Information Technology Industry Council	<a href="http://www.itic.org">http://www.itic.org</a>
ITU-T	International Telecommunications Union Telecommunications Standardization Sector	<a href="http://www.itu.int">http://www.itu.int</a>
JIS	Japanese Industrial Standards Committee	<a href="http://www.jisc.org">http://www.jisc.org</a>
T10	INCITS T10 Committee - SCSI storage interfaces	<a href="http://www.t10.org">http://www.t10.org</a>
T11	INCITS T11 Committee - Fibre Channel interfaces	<a href="http://www.t11.org">http://www.t11.org</a>
T13	INCITS T13 Committee - ATA storage interface	<a href="http://www.t13.org">http://www.t13.org</a>

### 2.2 Approved references

At the time of publication, the following referenced standards were approved.

ISO/IEC 14776-342, *SCSI-3 Controller Commands - 2 (SCC-2)*(ANSI INCITS 318-1998)

## 2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the documents, or regarding availability, contact the relevant standards body as indicated.

ISO/IEC 14776-414, *SCSI Architecture Model - 4 (SAM-4) standard* (T10/1638-D)  
ISO/IEC 14776-454, *SCSI Primary Commands - 4 (SPC-4) standard* (T10/1731-D)  
ISO/IEC 14776-352, *SCSI Media Changer Commands - 2 (SMC-2) standard* (T10/1383-D)  
ISO/IEC 14776-xxx, *SCSI Multimedia Commands - 6 (MMC-6) standard* (T10/1836-D)  
ISO/IEC 14776-372, *SCSI Enclosure Services - 2 (SES-2) standard* (T10/1559-D)

NOTE 1 - For more information on the current status of the document, contact the INCITS Secretariat at 202-737-8888 (telephone), 202-638-4922 (fax) or via Email at [incits@itic.org](mailto:incits@itic.org). To obtain copies of this document, contact Global Engineering at 15 Inverness Way East Englewood, CO 80112-5704 at 800-854-7179 (telephone), 303-792-2181 (telephone), or 303-792-2192 (fax).

### 3 Definitions, symbols, abbreviations, keywords, and conventions

#### 3.1 Definitions

**3.1.1 additional sense code:** A combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in the sense data. See SPC-4.

**3.1.2 application client:** An object that is the source of SCSI commands. See SAM-4.

**3.1.3 byte:** A sequence of eight contiguous bits considered as a unit.

**3.1.4 cache:** A temporary and often volatile data storage area outside the area accessible by application clients that may contain a subset of the data stored in the non-volatile data storage area.

**3.1.5 check data:** Information contained within a redundancy group (see 3.1.48) that may allow lost or destroyed XOR-protected data (see 3.1.61) to be recreated.

**3.1.6 command:** A request describing a unit of work to be performed by a device server. See SAM-4.

**3.1.7 command descriptor block (CDB):** The structure used to communicate commands from an application client to a device server. See SPC-4.

**3.1.8 cyclic redundancy check (CRC):** An error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum. See 4.17.4.

**3.1.9 data defect list (DLIST):** A list of defects sent by the application client to the device server during a FORMAT UNIT command. See 4.9.

**3.1.10 data-in buffer:** The buffer identified by the application client to receive data from the device server during the processing of a command. See SAM-4.

**3.1.11 data integrity field (DIF):** Another term for protection information (see 3.1.42).

**3.1.12 data-out buffer:** The buffer identified by the application client to supply data that is sent from the application client to the device server during the processing of a command. See SAM-4.

**3.1.13 default protection information:** Values placed into protection information fields if an application client does not specify specific protection information values.

**3.1.14 device server:** An object within a logical unit (see 3.1.30) that processes SCSI commands and some task management functions. See SAM-4.

**3.1.15 device type:** The type of device (or device model) implemented by the device server as indicated by the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data. See SPC-4.

**3.1.16 direct-access block device:** A device that is capable of containing data stored in logical blocks that each have a unique LBA. See 4.1.

**3.1.17 domain:** An I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem. See SAM-4.

**3.1.18 error correcting code (ECC):** An error checking mechanism that checks data integrity and enables some errors in the data to be corrected.

**3.1.19 exclusive-or (XOR):** A Boolean arithmetic function on two binary input values that results in an output value of 1 if one and only one of the input values is 1.



**3.1.20 extent:** A fixed set of logical blocks occupying contiguous LBAs on a single logical unit.

**3.1.21 field:** A group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.7) or sense data (see SPC-4).

**3.1.22 format corrupt:** a vendor-specific condition in which the application client may not be able to perform read operations, write operations, or verify operations. See 4.7.

**3.1.23 grown defect list (GLIST):** All defects sent by the application client to the device server. See 4.9.

**3.1.24 hard reset:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the hard reset operations described in SAM-4, this standard, and other applicable command standards (see table 13 in 5.1).

**3.1.25 I\_T nexus:** A relationship between a SCSI initiator port and a SCSI target port (see SAM-4).

**3.1.26 I\_T nexus loss:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the I\_T nexus loss operations described in SAM-4, this standard, and other applicable command standards (see table 13 in 5.1).

**3.1.27 logical block:** A set of data bytes accessed and referenced as a unit. See 4.4.

**3.1.28 logical block address (LBA):** The value used to reference a logical block (see 4.4).

**3.1.29 logical block length:** The number of bytes of user data in a logical block (see 4.4).

**3.1.30 logical unit:** An externally addressable entity within a SCSI target device (see 3.1.49) that implements a SCSI device model and contains a device server. See SAM-4.

**3.1.31 logical unit certification list (CLIST):** Defects detected by the device server during an optional certification process performed during the FORMAT UNIT command. See 4.9.

**3.1.32 logical unit reset:** A condition resulting from the events defined by SAM-4 in which the logical unit performs the logical unit reset operations described in SAM-4, this standard, and other applicable command standards (see table 13 in 5.1).

**3.1.33 media:** Plural of medium.

**3.1.34 medium:** The material on which data is stored (e.g., a magnetic disk).

**3.1.35 non-volatile cache:** Cache that retains data through power cycles.

**3.1.36 non-volatile medium:** A physical storage medium that retains data written to it for subsequent read operations through power cycles (e.g., a disk within a device that stores data as magnetic field changes that do not require device power to exist).

**3.1.37 physical block:** A set of data bytes accessed as a unit by the device server. See 4.5.

**3.1.38 physical block length:** The number of bytes of user data in a physical block (see 4.5).

**3.1.39 power cycle:** Power being removed followed by power being applied to a SCSI device.

**3.1.40 power on:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the power on operations described in SAM-4, this standard, and other applicable command standards (see table 13 in 5.1).

**3.1.41 primary defect list (PLIST):** The list of defects that are considered permanent defects. See 4.9.

**3.1.42 protection information:** Fields appended to each logical block that contain a cyclic redundancy check (CRC), an application tag, and a reference tag. See 4.17.

**3.1.43 pseudo read data:** Data that is transferred by a device server based on the setting of the RC bit or the TB bit in the Read-Write Error mode page (see 6.3.5) in order to maintain a continuous flow of data or to transfer the amount of data requested for a command even though an unrecovered read error (see 3.1.56) occurred while the device server was processing the command. Pseudo read data may be erroneous or fabricated by the device server (e.g., data already in a buffer or any other vendor-specific data).

**3.1.44 pseudo unrecovered error:** A simulated error (e.g., created by a WRITE LONG command (see 5.35 and 5.37)) for which a device server reports that it is unable to read or write a logical block, regardless of whether the data on the medium is valid, recoverable, or unrecoverable.

**3.1.45 pseudo unrecovered error with correction enabled:** A pseudo unrecovered error (see 3.1.44) for which a device server performs the maximum error recovery as specified in the Read-Write Error Recovery mode page (see 6.3.5). See 4.14.2.

**3.1.46 pseudo unrecovered error with correction disabled:** A pseudo unrecovered error (see 3.1.44) for which a device server performs no error recovery. See 4.14.2.

**3.1.47 recovered error:** An error for which a device server is able to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.3.5) and the Verify Error Recovery mode page (see 6.3.6).

**3.1.48 redundancy group:** A grouping of XOR-protected data (see 3.1.61) and associated check data (see 3.1.5) into a single type of data redundancy (see SCC-2). This standard only supports the XOR (see 3.1.19) type of redundancy.

**3.1.49 SCSI target device:** A SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices. See SAM-4.

**3.1.50 sense data:** Data describing an error or exceptional condition that a device server delivers to an application client in association with CHECK CONDITION status. See SPC-4.

**3.1.51 sense key:** The contents of the SENSE KEY field in the sense data. See SPC-4.

**3.1.52 status:** One byte of response information sent from a device server to an application client upon completion of each command. See SAM-4.

**3.1.53 storage array controller:** Any combination of an initiator and application clients (see SAM-4) that originates SCSI commands, converts input LUNs to output LUNs, and converts input LBAs to output LBAs. A storage array controller organizes a group of direct-access block devices into various objects (e.g., redundancy groups and volume sets). See SCC-2.

**3.1.54 unit attention condition:** A state that a logical unit (see 3.1.30) maintains while the logical unit has asynchronous status information to report to the initiator ports associated with one or more I\_T nexuses (see 3.1.25). See SAM-4.

**3.1.55 unrecovered error:** An error for which a device server is unable to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.3.5) and the Verify Error Recovery mode page (see 6.3.6).

**3.1.56 unrecovered read error:** An unrecovered error (see 3.1.55) during a read operation.

**3.1.57 user data:** Data contained in logical blocks that is not protection information.

**3.1.58 volatile cache:** Cache that does not retain data through power cycles.

**3.1.59 volatile medium:** Medium that does not retain data written to it for a subsequent read operation through power cycles (e.g., a silicon memory device that loses data written to it if device power is lost).

**3.1.60 XOR operation:** Performing an XOR (see 3.1.19) bitwise on two identical-sized multiple-bit input values (e.g., the current value of a logical block and the new value for that logical block). In a storage array implementing a redundancy group (see 3.1.48), the XOR operation is used in error correction algorithms and may be performed by the storage array controller (see 3.1.53) or by the direct-access block devices (see 3.1.16). See 4.15.

**3.1.61 XOR-protected data:** Logical blocks, including user data and protection information, if any, that are part of a redundancy group (see 3.1.48).

## 3.2 Symbols and abbreviations

See table 1 for abbreviations of standards bodies (e.g., ISO). Additional symbols and abbreviations used in this standard include:

Abbreviation	Meaning
CDB	command descriptor block (see 3.1.7)
CRC	cyclic redundancy check (see 3.1.8)
CLIST	logical unit certification list (see 3.1.31)
DIF	data integrity field (see 3.1.11)
DLIST	data defect list (see 3.1.9)
ECC	error correcting code (see 3.1.18)
GLIST	grown defect list (see 3.1.23)
I/O	input/output
LBA	logical block address (see 3.1.28)
LSB	least significant bit
LUN	logical unit number
MMC-6	SCSI Multimedia Commands - 6 standard
MSB	most significant bit
PLIST	primary defect list (see 3.1.41)
SAM-4	SCSI Architecture Model - 4 standard
SCSI	Small Computer System Interface family of standards
SCC-2	SCSI-3 Controller Commands - 2 standard
SES-2	SCSI Enclosure Services - 2 standard
SMC-2	SCSI Media Changer Commands - 2 standard
SPC-4	SCSI Primary Commands - 4 standard
XOR	exclusive-or (see 3.1.19)

## 3.3 Keywords

**3.3.1 expected:** A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

**3.3.2 ignored:** A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device.

**3.3.3 invalid:** A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

**3.3.4 mandatory:** A keyword indicating an item that is required to be implemented as defined in this standard.

**3.3.5 may:** A keyword that indicates flexibility of choice with no implied preference. "May" is equivalent to "may or may not".

**3.3.6 may not:** Keywords that indicate flexibility of choice with no implied preference. "May not" is equivalent to "may or may not".

**3.3.7 need not:** Keywords indicating a feature that is not required to be implemented. "Need not" is equivalent to "is not required to".

**3.3.8 obsolete:** A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

**3.3.9 optional:** A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined in this standard is implemented, then it shall be implemented as defined in this standard.

**3.3.10 prohibited:** A keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard.

**3.3.11 reserved:** A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

**3.3.12 restricted:** A keyword referring to bits, bytes, words, and fields that are set aside for use in other SCSI standards. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

**3.3.13 shall:** A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

**3.3.14 should:** A keyword indicating flexibility of choice with a strongly preferred alternative. "Should" is equivalent to the phrase "it is strongly recommended."

**3.3.15 vendor-specific:** Something (e.g., a bit, field, or code value) that is not defined by this standard and may be used differently in various implementations.

## 3.4 Conventions

### 3.4.1 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in this clause or in the text where they first appear.

Names of commands, status codes, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE).

Names of fields and state variables are in small uppercase (e.g. NAME). When a field or state variable name contains acronyms, uppercase letters may be used for readability. Normal case is used when the contents of

a field or state variable are being discussed. Fields or state variables containing only one bit are usually referred to as the NAME bit instead of the NAME field.

Normal case is used for words having the normal English meaning.

Notes do not constitute any requirements for implementers.

### 3.4.2 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores are included between characters in binary number representations to increase readability or delineate field boundaries (e.g., 0\_0101\_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores are included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B\_FD8C\_FA23h).

A decimal number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

This standard uses the following conventions for representing decimal numbers:

- a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
- b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space; and
- c) the thousands separator is used in both the integer portion and the fraction portion of a number.

Table 2 shows some examples of decimal numbers represented using various conventions.

**Table 2 — Numbering convention examples**

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666. $\overline{6}$  means 666.666 666... or 666  $\frac{2}{3}$ , and 12.142  $\overline{857}$  means 12.142 857 142 857... or 12  $\frac{1}{7}$ ).

A range of numeric values may be represented in this standard in the form “a to z”, where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation “0h to 3h” includes the values 0h, 1h, 2h, and 3h).

### 3.4.3 Lists conventions

#### 3.4.3.1 Lists conventions overview

Lists are introduced by a complete grammatical proposition followed by a colon and completed by the items in the list.

Each item in a list is preceded by an identification with the style of the identification being determined by whether the list is an ordered list or an unordered list.

If the item in a list is not a complete sentence, then the first word in the item is capitalized. If the item in a list is a complete sentence, then the first word in the item is capitalized,

Each item in a list ends with a semicolon, except the last item, which ends in a period. The next to the last entry in a list ends with a semicolon followed by an “and” or an “or” (i.e., “...; and”, or “...; or”). The “and” is used if all the items in the list are required. The “or” is used if only one or more items in the list are required.

#### **3.4.3.2 Unordered lists**

An unordered list is one in which the order of the listed items is unimportant (i.e., it does not matter where in the list an item occurs as all items have equal importance). Each list item starts with a lower case letter or an uppercase letter followed by a close parenthesis. The following is an example of an unordered list in which there is no ordered relationship between the colors named:

- a) red (i.e., one of the following colors):
  - A) crimson; or
  - B) amber;
- b) blue; or
- c) green.

#### **3.4.3.3 Ordered lists**

An ordered list is one in which the order of the listed items is important (i.e., item n is required before item n+1). Each listed item starts with an Western-Arabic numeral followed by a close parenthesis. The following is an example of an ordered list in which the order by which a page is to be read is specified:

- 1) top of the page;
- 2) middle of the page; and
- 3) bottom of the page.

#### **3.4.4 Precedence**

In the event of conflicting information the precedence for requirements defined in this standard is:

- 1) text;
- 2) tables; then
- 3) figures.

## 4 Direct-access block device type model

### 4.1 Direct-access block device type model overview

SCSI devices that conform to this standard are referred to as direct-access block devices. This includes the category of logical units commonly referred to as rigid disks and removable rigid disks. MMC-4 is typically used by CD-ROM devices.

This standard is intended to be used in conjunction with SAM-4, SPC-4, SCC-2, SES-2, and SMC-2.

Direct-access block devices store data for later retrieval in logical blocks. Logical blocks contain user data, may contain protection information accessible to the application client, and may contain additional information not normally accessible to the application client (e.g., an ECC). The number of bytes of user data contained in each logical block is the logical block length. The logical block length is greater than or equal to one byte and should be even. Most direct-access block devices support a logical block length of 512 bytes and some support additional logical block lengths (e.g., 520 or 4096 bytes). The logical block length does not include the length of protection information and additional information, if any, that are associated with the logical block. The logical block length is the same for all logical blocks on the medium.

Each logical block is stored at a unique LBA, which is either four bytes (i.e., a short LBA) or eight bytes (i.e., a long LBA) in length. The LBAs on a logical unit shall begin with zero and shall be contiguous up to the last logical block on the logical unit. An application client uses commands performing write operations to store logical blocks and commands performing read operations to retrieve logical blocks. A write operation causes one or more logical blocks to be written to the medium. A read operation causes one or more logical blocks to be read from the medium. A verify operation confirms that one or more logical blocks were correctly written and are able to be read without error from the medium.

Logical blocks are stored by a process that causes localized changes or transitions within a medium. The changes made to the medium to store the logical blocks may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles). The medium may contain vendor-specific information that is not addressable through an LBA. Such data may include defect management data and other device management information.

### 4.2 Media examples

#### 4.2.1 Media examples overview

Examples of types of media used by the direct-access block device are:

- a) rotating media (see 4.2.2); and
- b) memory media (see 4.2.3).

Other types of media are possible.

#### 4.2.2 Rotating media

The typical application of a direct-access block device is a magnetic disk device. The medium is a spinning disk with a magnetic material that allows flux changes to be induced and recorded. An actuator positions a read-write head radially across the spinning disk, allowing the device to randomly read or write the information at any radial position. Data is stored by using the write portion of the head to record flux changes and is read by using the read portion of the head to read the recorded data.

The circular path followed by the read-write head at a particular radius is called a track. The track is divided into sectors each containing blocks of stored data. If there are more than one disk spinning on a single axis and the actuator has one or more read-write heads to access the disk surfaces, the collection of tracks at a particular radius is called a cylinder.

A logical block is stored in one or more sectors, or a sector may store more than one logical block. Sectors may also contain information for accessing, synchronizing, and protecting the integrity of the logical blocks.

A rotating media-based direct-access block device is ready when the disks are rotating at the correct speed and the read-write circuitry is powered and ready to access the data, and may require a START STOP UNIT command (see 5.19) to bring the logical unit to the ready state.

Rotating media-based direct-access block device are usually non-volatile.

The defect management scheme of a disk device may not be discernible through this command set, though some aspects (see 4.9) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.16, 5.17, 5.35, and 5.36).

#### **4.2.3 Memory media**

Memory media is based on solid state random access memories (RAMs) (e.g., static RAM (SRAM), dynamic RAM (DRAM), magnetoresistive RAM (MRAM), ferroelectric RAM (FeRAM), or flash memory). Memory media-based direct-access block devices may be used for fast-access storage.

A memory media-based direct-access block device is ready after power on, and does not require a START STOP UNIT command (see 5.19) to bring the logical unit to a ready state.

These logical units may be non-mechanical, and therefore logical blocks may be accessed with similar access times regardless of their location on the medium. Memory media-based direct-access block devices may store less data than disks or tapes, and may be volatile.

The defect management scheme (e.g., ECC bytes) (see 4.9) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.16, 5.17, 5.35, and 5.36).

Memory media may be volatile (e.g., SRAM or DRAM) or non-volatile (e.g., SRAM or DRAM with battery backup, MRAM, FeRAM, or flash memory).

### **4.3 Removable medium**

#### **4.3.1 Removable medium overview**

The medium may be removable or non-removable. The removable medium may be contained within a cartridge or jacket to prevent damage to the recording surfaces.

A removable medium has an attribute of being mounted or unmounted on a suitable transport mechanism in a direct-access block device. A removable medium is mounted when the direct-access block device is capable of performing write, read, and verify operations to the medium. A removable medium is unmounted at any other time (e.g., during loading, unloading, or storage).

An application client may check whether a removable medium is mounted by issuing a TEST UNIT READY command (see SPC-4). A direct-access block device containing a removable medium may not be accessible for write, read, and verify operations until it receives a START STOP UNIT command (see 5.19).

If the direct-access block device implements cache, either volatile or non-volatile, it ensures that all logical blocks of the medium contain the most recent user data and protection information, if any, prior to permitting unmounting of the removable medium.

If the medium in a direct-access block device is removable, and the medium is removed, then the device server shall establish a unit attention condition with the additional sense code set to the appropriate value (e.g., NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED).

The PREVENT ALLOW MEDIUM REMOVAL command (see 5.6) allows an application client to restrict the unmounting of the removable medium. This is useful in maintaining system integrity.

If the application client issues a START STOP UNIT command to eject the removable medium, and the direct-access block device is prevented from unmounting by the PREVENT ALLOW MEDIUM REMOVAL command, the START STOP UNIT command is rejected by the device server.



### 4.3.2 Removable medium with an attached media changer

When a direct-access block device is served by an attached media changer, control over a medium transport element may be accomplished using media changer commands (see SMC-2) sent to the direct-access block device type logical unit.

The direct-access block device indicates its ability to support these commands by setting the MCHNGR bit to one in its standard INQUIRY data (see SPC-4). A MCHNGR bit set to one indicates that the MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED commands (see SMC-2) are supported. The logical unit may require a MODE MEDIUM ATTACHED command (see SMC-2) to become ready.

## 4.4 Logical blocks

Logical blocks are stored on the medium along with additional information that the device server uses to manage storage and retrieval. The format of the additional information is defined by other standards or is vendor-specific and is hidden from the application client during normal read, write, and verify operations. This additional information may be used to identify the physical location of the blocks of data, the address of the logical block, and to provide protection against the loss of user data and protection information, if any (e.g., by containing ECC bytes).

The first LBA is zero. The last LBA is  $[n-1]$ , where  $[n]$  is the number of logical blocks on the medium accessible by the application client. The READ CAPACITY (10) parameter data (see 5.12.2 and 5.13.2) RETURNED LOGICAL BLOCK ADDRESS field indicates the value of  $[n-1]$ .

LBAs are no larger than 8 bytes. Some commands support only 4-byte (i.e., short) LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (10), READ (10), and WRITE (10)). If the capacity exceeds that accessible with short LBAs, then the device server returns a capacity of FFFF\_FFFFh in response to a READ CAPACITY (10) command, indicating that:

- a) the application client should enable descriptor format sense data (see SPC-4) in the Control mode page (see SPC-4) and in any REQUEST SENSE commands (see SPC-4) it sends; and
- b) the application client should use commands with 8-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (16), READ (16), and WRITE (16)).

NOTE 2 - If a command with a 4-byte LOGICAL BLOCK ADDRESS field accesses logical blocks beyond LBAs FFFF\_FFFFh and fixed format sense data is used, there is no field in the sense data large enough to report the LBA of an error (see 4.14).

If a command is received that references or attempts to access a logical block not within the capacity of the medium, then the device server terminates the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server may terminate the command before processing or after the device server has transferred some or all of the data.

The number of bytes of user data contained in a logical block is the logical block length. The parameter data returned by the device server in response to a READ CAPACITY command (see 5.12) describes the logical block length that is used on the medium. The mode parameter block descriptor (see 6.3.2) is used by an application client to change the logical block length in direct-access block devices that support changeable logical block lengths. The logical block length should be used to determine does not include the length of protection information and additional information, if any.

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical direct-access block device, the time to access a logical block at LBA  $[x+1]$  after accessing LBA  $[x]$  is often less than the time to access some other logical block. The time to access the logical block at LBA  $[x]$  and then the logical block at LBA  $[x+1]$  need not be less than time to access LBA  $[x]$  and then LBA  $[x+100]$ . The READ CAPACITY command issued with a PMI bit set to one may be useful in determining where longer access times occur.

## 4.5 Physical blocks

A physical block is a set of data bytes on the medium accessed by the device server as a unit. A physical block may contain:

- a) a portion of a logical block (i.e., there are multiple physical blocks in the logical block)(e.g., a physical block length of 512 bytes with a logical block length of 2 048 bytes);
- b) a single complete logical block; or
- c) more than one logical block (i.e., there are multiple logical blocks in the physical block)(e.g., a physical block length of 4 096 bytes with a logical block length of 512 bytes).

Each physical block includes additional information not normally accessible to the application client (e.g., an ECC) that the device server uses to manage storage and retrieval.

If the device server supports the COR\_DIS bit and/or the WR\_UNCOR bit in a WRITE LONG command (see 5.35 and 5.36), then the device server shall have the capability of marking individual logical blocks as containing pseudo uncorrectable errors with correction enabled (see 3.1.45) or with correction disabled (see 3.1.46).

Logical blocks may or may not be aligned to physical block boundaries. A mechanism for establishing the alignment is not defined by this standard.

Figure 2 shows examples of logical blocks and physical blocks, where LBA 0 is aligned to a physical block boundary.

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 0h  
(indicating one or more physical blocks per logical block):

4 physical blocks per logical block:

LBA 0				LBA 1				...
PB	PB	PB	PB	PB	PB	PB	PB	...

3 physical blocks per logical block:

LBA 0			LBA 1			LBA 2			...
PB	PB	PB	PB	PB	PB	PB	PB	PB	...

2 physical blocks per logical block:

LBA 0		LBA 1		LBA 2		LBA 3		LBA 4		...
PB	PB	PB	PB	PB	PB	PB	PB	PB	PB	...

1 physical block per logical block:

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	LBA 10	...
PB	PB	PB	PB	PB	PB	PB	PB	PB	PB	PB	...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to a non-zero value  
(indicating more than one logical block per physical block):

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 1h (indicating  $2^1$  logical blocks per physical block):

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	LBA 10	LBA 11	...
PB		PB		PB		PB		PB		PB		...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating  $2^2$  logical blocks per physical block):

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	LBA 10	LBA 11	...
PB				PB				PB				...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 3h (indicating  $2^3$  logical blocks per physical block):

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	...
PB								...

**Key:**

LBA n = logical block with LBA n

PB = physical block

**Figure 2 — Logical blocks and physical blocks examples**

Figure 3 shows examples of logical blocks and physical blocks, where various LBAs are aligned to the physical block boundaries.

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 1h (indicating  $2^1$  logical blocks per physical block):

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0:

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	...
PB		PB		PB		PB		PB		...

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 1:

	LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	LBA 10	...
PB		PB		PB		PB		PB		PB		...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating  $2^2$  logical blocks per physical block):

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0:

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	...
PB				PB				...

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 1:

		LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	...
PB			PB			PB			PB		...

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 2:

	LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	...
PB			PB			PB			...		

LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 3:

				LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7	LBA 8	LBA 9	LBA 10	...
PB				PB				PB				PB			...

**Key:**

LBA n = logical block with LBA n

PB = physical block

**Figure 3 — Logical block to physical block alignment examples**

When there are more than one logical block per physical block, not all of the logical blocks are aligned to the physical block boundaries. When using medium access commands, application clients should:

- specify an LBA that is aligned to a physical block boundary; and
- access an integral number of physical blocks, provided that the access does not go beyond the last LBA on the medium.

## 4.6 Ready state

A direct-access block device is ready when the device server is capable of processing medium access commands (i.e., commands that perform read operations, write operations, or verify operations).

A direct-access block device using removable media is not ready until a volume is mounted and other conditions are met (see 4.2). If a direct-access block device is not ready, then the device server shall

terminate medium access commands with CHECK CONDITION status with the sense key set to NOT READY and the appropriate additional sense code for the condition.

Some direct-access block devices may be switched from being ready to being not ready by using the START STOP UNIT command (see 5.19). An application client may need to issue a START STOP UNIT command with a START bit set to one to make a direct-access block device ready.

## 4.7 Initialization

Direct-access block devices may require initialization prior to write, read, and verify operations. This initialization is performed by a FORMAT UNIT command (see 5.2). Parameters related to the format (e.g., logical block length) may be set with the MODE SELECT command prior to the format operation. Some direct-access block devices are initialized by means not specified in this standard. The time when the initialization occurs is vendor-specific.

Direct-access block devices using a non-volatile medium may save the parameters and only need to be initialized once. However, some mode parameters may need to be initialized after each logical unit reset. A catastrophic failure of the direct-access block device may require the FORMAT UNIT command to be issued.

Direct-access block devices that use a volatile medium may need to be initialized after each logical unit reset prior to the processing of write, read, or verify operations. Mode parameters may also need initialization after logical unit resets.

NOTE 3 - Mode parameter block descriptors read with the MODE SENSE command before a FORMAT UNIT completes may contain information that may not reflect the true state of the medium.

A direct-access block device may become format corrupt after processing a MODE SELECT command that changes parameters related to the medium format. During this time, the device server may terminate medium access commands with CHECK CONDITION status with the sense key set to NOT READY and the appropriate additional sense code for the condition.

Any time the parameter data to be returned by a device server in response to a READ CAPACITY (10) command (see 5.12) or a READ CAPACITY (16) command (see 5.13) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command completes changing the number of logical blocks, logical block length, protection information, or reference tag ownership values, or when a vendor-specific mechanism causes a change), then the device server shall establish a unit attention condition for the SCSI initiator port (see SAM-4) associated with each L\_T nexus, except the L\_T nexus on which the command causing the change was received, with the additional sense code set to CAPACITY DATA HAS CHANGED.

NOTE 4 - Logical units compliant with previous versions of this standard were not required to establish a unit attention condition.

## 4.8 Write protection

Write protection prevents the alteration of the medium by commands issued to the device server. Write protection is usually controlled by the user of the medium through manual intervention (e.g., mechanical lock) or may result from hardware controls (e.g., tabs on the media housing) or software write protection. All sources of write protection are independent. When present, any write protection shall cause otherwise valid commands that request alteration of the medium to be rejected with CHECK CONDITION status with the sense key set to DATA PROTECT. Only when all write protections are disabled shall the device server process commands that request alteration of the medium.

Hardware write protection results when a physical attribute of the drive or medium is changed to specify that writing shall be prohibited. Changing the state of the hardware write protection requires physical intervention, either with the drive or the medium. If allowed by the drive, changing the hardware write protection while the medium is mounted results in vendor-specific behavior that may include the writing of previously buffered data (e.g., data in cache).

Software write protection results when the device server is marked as write protected by the application client using the SWP bit in the Control mode page (see SPC-4). Software write protection is optional. Changing the

state of software write protection shall not prevent previously accepted data (e.g., data in cache) from being written to the media.

The device server reports the status of write protection in the device server and on the medium with the `DEVICE-SPECIFIC PARAMETER` field in the mode parameter header (see 6.3.1).

## 4.9 Medium defects

Any medium has the potential for defects that cause data to be lost. Therefore, each logical block may contain additional information that allows the detection of changes to the user data and protection information, if any, caused by defects in the medium or other phenomena, and may also allow the data to be reconstructed following the detection of such a change (e.g., ECC bytes).

A defect causes a recovered error (see 3.1.47) if the device server is able to retrieve the data by retrying or using the additional information to reconstruct the data. A defect causes an unrecovered error (see 3.1.55) if the device server is unable to retrieve the data.

Direct-access block devices may allow the application client to examine and modify the additional information by using the `READ LONG` commands and the `WRITE LONG` commands (see 5.16, 5.17, 5.35, and 5.36). The application client may use the `WRITE LONG` commands to alter the additional information to test the defect detection logic of the direct-access block device or to emulate a logical block with an unrecovered read error when generating a mirror copy. This may induce a recovered error or an unrecovered error.

Direct-access block devices may allow the application client to use the features of the `WRITE LONG` commands (see 5.35 and 5.36) to:

- a) disable error correction on specific logical blocks or physical blocks;
- b) disable automatic reallocation on specific logical blocks or physical blocks; and
- c) mark specific logical blocks or physical blocks as containing pseudo unrecoverable errors with correction enabled or with correction disabled.

These features provide methods for an application client to prevent logical blocks from being reported as information exception conditions and unnecessary reassignments.

During a self-test operation (see SPC-4), the device server shall ignore pseudo unrecoverable errors with correction disabled and shall process the pseudo unrecoverable errors with correction enabled. See 4.19.1 for the rules for background scans.

Defects may also be detected and managed during processing of a `FORMAT UNIT` command (see 5.2). The `FORMAT UNIT` command defines four sources of defect information: the `PLIST`, `CLIST`, `DLIST`, and `GLIST`. These defects may be reassigned or avoided during the initialization process so that they do not affect any logical blocks. The sources of defect location information (i.e., defects) are defined as follows:

- a) Primary defect list (`PLIST`). This is the list of defects, which may be supplied by the original manufacturer of the device or medium, that are considered permanent defects. The `PLIST` is located outside of the application client accessible logical block space. The `PLIST` is accessible by the device server for reference during the format operation, but it is not accessible by the application client except through the `READ DEFECT DATA` commands (see 5.12 and 5.15). Once created, the original `PLIST` shall not change;
- b) Logical unit certification list (`CLIST`). This list includes defects detected by the device server during an optional certification process performed during the `FORMAT UNIT` command. This list shall be added to the `GLIST`;
- c) Data defect list (`DLIST`). This list of defects may be supplied by the application client to the device server during the `FORMAT UNIT` command. This list shall be added to the `GLIST`; and
- d) Grown defect list (`GLIST`). The `GLIST` includes all defects sent by the application client (i.e., the `DLIST`) or detected by the device server (i.e., the `CLIST`). The `GLIST` does not include the `PLIST`. If the `CMPLST` bit is set to zero, the `GLIST` shall include `DLIST`s provided to the device server during the previous and the current `FORMAT UNIT` commands. The `GLIST` shall also include:
  - A) defects detected by the format operation during medium certification;
  - B) defects previously identified with a `REASSIGN BLOCKS` command (see 5.18); and
  - C) defects previously detected by the device server and automatically reallocated.

The direct-access block device may automatically reassign defects if allowed by the Read-Write Error Recovery mode page (see 6.3.5).

Defects may also occur after initialization. The application client issues a REASSIGN BLOCKS command (see 5.18) to request that the specified LBA be reassigned to a different part of the medium. This operation may be repeated if a new defect appears at a later time. The total number of defects that may be handled in this manner is vendor-specific.

Defect management on direct-access block devices is vendor-specific. Direct-access block devices not using a removable medium may optimize the defect management for capacity or performance or both. Some direct-access block devices that use a removable medium do not support defect management or use defect management that does not impede the ability to interchange the medium.

## 4.10 Write failures

If one or more commands performing write operations are in the task set and are being processed when power is lost (e.g., resulting in a vendor-specific command timeout by the application client) or a medium error or hardware error occurs (e.g., because a removable medium was incorrectly unmounted), then the data in the logical blocks being written by those commands is indeterminate. When accessed by a command performing a read or verify operation (e.g., after power on or after the removable medium is mounted), the device server may return old data, new data, or vendor-specific data in those logical blocks.

Before reading logical blocks which encountered such a failure, an application client should reissue any commands performing write operations that were outstanding.

## 4.11 Caches

Direct-access block devices may implement caches. A cache is an area of temporary storage in the direct-access block device with a fast access time that is used to enhance performance. Cache exists separately from the medium and is not directly accessible by the application client. Use of cache for write or read operations may reduce the access time to a logical block and increase the overall data throughput.

Cache stores user data and protection information, if any.

Cache may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Non-volatile cache memories retain data through power cycles. There may be a limit on the amount of time a non-volatile cache is able to retain data without power.

The power, if any, that allows a non-volatile cache to remain non-volatile may become low enough to prevent the non-volatile cache from remaining non-volatile for a vendor specific minimum time (e.g., the battery voltage becomes too low to sustain cache contents beyond a vendor specific time). If this occurs and the Extended INQUIRY Data VPD page (see SPC-4) indicates that the device server contains non-volatile cache (i.e., NV\_SUP bit set to one), then:

- a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see SPC-4)), then the device server shall report the degraded non-volatile cache as specified in the Information Exceptions Control mode page with an additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE; or
- b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus with the additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE.

Non-volatile caches may become volatile (e.g., battery voltage becomes too low to sustain cache contents when power is lost). In this case, read or write operations requested by commands in which the force unit access non-volatile cache (FUA\_NV) bit in the CDB is set to one may bypass the cache resulting in a decrease in overall data throughput.

If a non-volatile cache becomes volatile, then the device server shall set the REMAINING NON-VOLATILE TIME field to zero in the Non-volatile Cache log page (see 6.2.4).

If non-volatile cache becomes volatile and the Extended INQUIRY Data VPD page (see SPC-4) indicates that the device server contains non-volatile cache (i.e., the NV\_SUP bit is set to one) then:

- a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see SPC-4)), then the device server shall report the change in the cache as specified in the Information Exceptions Control mode page with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE; or
- b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE.

If:

- a) a power on or hard reset occurs;
- b) the Extended INQUIRY Data VPD page indicates that the device server contains a non-volatile cache (i.e., the NV\_SUP bit set to one); and
- c) the non-volatile cache is currently volatile,

then the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE.

During read operations, the device server uses the cache to store logical blocks that the application client may request at some future time. The algorithm used to manage the cache is not part of this standard. However, parameters are provided to advise the device server about future requests, or to restrict the use of cache for a particular request.

During write operations, the device server uses the cache to store data that is to be written to the medium at a later time. This is called write-back caching. The command may complete prior to logical blocks being written to the medium. As a result of using a write-back caching there is a period of time when the data may be lost if power to the SCSI target device is lost and a volatile cache is being used or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write operation. If an error occurred during the write operation, it may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled with the Caching mode page (see 6.3.4) to prevent detected write errors from being reported as deferred errors. Even with write-back caching disabled, undetected write errors may occur. The VERIFY commands and the WRITE AND VERIFY commands may be used to detect those errors.

When the cache becomes full of logical blocks, new logical blocks may replace those currently in the cache. The disable page out (DPO) bit in the CDB of commands performing write, read, or verify operations allows the application client to influence the replacement of logical blocks in the cache. For write operations, setting the DPO bit to one specifies that the device server should not replace existing logical blocks in the cache with the new logical blocks being written. For read and verify operations, setting the DPO bit to one specifies that the device server should not replace logical blocks in the cache with the logical blocks that are being read.

NOTE 5 - This does not mean that stale data is allowed in the cache. If a write operation accesses the same LBA as a logical block in the cache, the logical block in the cache is updated with the new write data.

Application clients may use the force unit access (FUA) bit in the CDB of commands performing write or read operations to specify that the device server shall access the medium. For a write operation, setting the FUA bit to one causes the device server to complete the data write to the medium before completing the command. For a read operation, setting the FUA bit to one causes the device server to retrieve the logical blocks from the medium rather than from the cache.

When the DPO and FUA bits are both set to one, write and read operations effectively bypass the cache.

Application clients may use the force unit access non-volatile cache (FUA\_NV) bit in the CDB of commands performing write or read operations to specify that the device server may access a non-volatile cache, if any, rather than the medium, if the FUA bit is set to zero. For a write operation, an FUA\_NV bit set to one with the FUA bit set to zero allows the device server to complete the data write to non-volatile cache rather than the medium before completing the command. For a read operation, an FUA\_NV bit set to one with the FUA bit set to zero allows the device server to retrieve the logical blocks from the non-volatile cache rather than the medium.



When a VERIFY command or a WRITE AND VERIFY command is processed, both a force unit access and a synchronize cache operation are implied, since the logical blocks are being verified as being stored on the medium. The DPO bit is defined in the VERIFY command since the VERIFY command may cause the replacement of logical blocks in the cache.

Commands may be implemented by the device server that allow the application client to control other behavior of the cache:

- a) the PRE-FETCH commands (see 5.4 and 5.5) cause a set of logical blocks requested by the application client to be read into cache for possible future access. The logical blocks fetched are subject to later replacement;
- b) the SYNCHRONIZE CACHE commands (see 5.20 and 5.21) force any write data in cache in the requested set of logical blocks to be written to the medium. These commands may be used to ensure that the data is written and any detected errors reported;
- c) the Caching mode page (see 6.3.4) provides control of cache behavior.

#### 4.12 Implicit HEAD OF QUEUE command processing

Each of the following commands defined by this standard may be processed by the task manager as if it has a HEAD OF QUEUE task attribute (see SAM-4), even if the command is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

- a) the READ CAPACITY (10) command; and
- b) the READ CAPACITY (16) command.

See SPC-4 for additional commands subject to implicit HEAD OF QUEUE command processing.

Application clients should not send a command with the ORDERED task attribute if the command may be processed as if it has a HEAD OF QUEUE task attribute of, because whether the ORDERED task attribute is processed is vendor-specific.

#### 4.13 Reservations

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-4 for a description of reservations. The details of commands that are allowed under what types of reservations are described in table 3.

Commands from I\_T nexuses holding a reservation should complete normally. Table 3 specifies the behavior of commands from registered I\_T nexuses when a registrants only or all registrants type persistent reservation is present.

For each command, this standard or SPC-4 defines the conditions that result in the device server completing the command with RESERVATION CONFLICT status.

Table 3 — SBC-2 commands that are allowed in the presence of various reservations

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From I_T nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
FORMAT UNIT	Conflict	Conflict	Allowed	Conflict	Conflict
ORWRITE	Conflict	Conflict	Allowed	Conflict	Conflict
PRE-FETCH (10)/(16)	Allowed	Conflict	Allowed	Allowed	Conflict
PREVENT ALLOW MEDIUM REMOVAL (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT ALLOW MEDIUM REMOVAL (Prevent<>0)	Conflict	Conflict	Allowed	Conflict	Conflict
READ (6)/(10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
READ CAPACITY (10)/(16)	Allowed	Allowed	Allowed	Allowed	Allowed
READ DEFECT DATA (10)/(12)	Allowed <sup>a</sup>	Conflict	Allowed	Allowed <sup>a</sup>	Conflict
READ LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
REASSIGN BLOCKS	Conflict	Conflict	Allowed	Conflict	Conflict
START STOP UNIT with START bit set to one and POWER CONDITION field set to 0h	Allowed	Allowed	Allowed	Allowed	Allowed
START STOP UNIT with START bit set to zero or POWER CONDITION field set to a value other than 0h	Conflict	Conflict	Allowed	Conflict	Conflict
SYNCHRONIZE CACHE (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
VERIFY (10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE (6)/(10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE AND VERIFY (10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SAME (10)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDREAD (10)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
XDWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDWRITEREAD (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XPWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
<p><b>Key:</b> RR = Registrants Only or All Registrants</p> <p><b>Allowed:</b> Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.</p> <p><b>Conflict:</b> Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed, and the device server shall complete the command with RESERVATION CONFLICT status.</p> <p><sup>a</sup> The device server in logical units claiming compliance with previous versions of this standard (e.g., SBC-2) may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-4).</p>					

## 4.14 Error reporting

### 4.14.1 Error reporting overview

If any of the conditions listed in table 4 occur during the processing of a command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to the specified value and the additional sense code set to the appropriate value for the condition. Some errors may occur after the completion status has already been reported. For such errors, SPC-4 defines a deferred error reporting mechanism. Table 4 lists some error conditions and the applicable sense keys. The list does not provide a complete list of all conditions that may cause CHECK CONDITION status.

**Table 4 — Example error conditions**

Condition	Sense key
Invalid LBA	ILLEGAL REQUEST
Unsupported option requested	ILLEGAL REQUEST
Logical unit reset, I_T nexus loss, or medium change since last command from this application client	UNIT ATTENTION
Self diagnostic failed	HARDWARE ERROR
Unrecovered error	MEDIUM ERROR or HARDWARE ERROR
Recovered read error	RECOVERED ERROR
Pseudo recovered error	MEDIUM ERROR
Over-run or other error that might be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write-protected medium	DATA PROTECT

Direct-access block devices compliant with this standard shall support both the fixed and descriptor formats of sense data (see SPC-4). If fixed format sense data is used but the values to be placed in the sense data INFORMATION field or COMMAND-SPECIFIC INFORMATION field are too large for the fixed format sense data (e.g., an 8-byte LBA), the VALID bit shall be set to zero.

Table 5 summarizes use of the sense data fields.

**Table 5 — Sense data field usage for direct-access block devices**

Field	Usage	Reference
VALID bit and INFORMATION field	READ LONG commands	5.16 and 5.17
	REASSIGN BLOCKS command	5.18
	WRITE LONG commands	5.35 and 5.36
	Any command that accesses the medium, based on the Read-Write Error Recovery mode page	6.3.5
COMMAND-SPECIFIC INFORMATION field	EXTENDED COPY command	SPC-4
	REASSIGN BLOCKS command	5.18
ILI bit	READ LONG commands	5.16 and 5.17
	WRITE LONG commands	5.35 and 5.36

When a command attempts to access or reference an invalid LBA, the device server shall return the first invalid LBA in the INFORMATION field of the sense data (see SPC-4).

When a recovered read error is reported, the INFORMATION field of the sense data shall contain the LBA of the last logical block accessed by the command on which a recovered read error occurred.

When an unrecovered error is reported, the INFORMATION field of the sense data shall contain the LBA of the logical block on which the unrecovered error occurred.

#### 4.14.2 Processing pseudo unrecovered errors

If a pseudo unrecovered error is encountered on a logical block with correction disabled (e.g., by a command, a background scan, or a background self-test), then the device server shall:

- a) perform no error recovery on the affected logical blocks, including any read error recovery enabled by the Read-Write Error Recovery mode page (see 6.3.5) or the Verify Error Recovery mode page (see 6.3.6);
- b) perform no automatic reallocation of the affected logical blocks, including any automatic reallocation enabled by the Read-Write Error Recovery mode page;
- c) not consider errors on the affected logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see SPC-4);
- d) not log errors on the affected logical blocks in the Error Counter log pages (see SPC-4); and
- e) in any information returned for the error (e.g., in sense data or in the Background Scan Results log page (see 6.2.2)), set the sense key set to MEDIUM ERROR and either:
  - A) should set the additional sense code to READ ERROR – LBA MARKED BAD BY APPLICATION CLIENT; or
  - B) may set the additional sense code to UNRECOVERABLE READ ERROR.

If a pseudo unrecovered error is encountered on a logical block with correction enabled (e.g., by a command, a background scan, or a background self-test), the device server shall:

- a) if enabled by the Read-Write Error Recovery mode page (see 6.3.5) or the Verify Error Recovery mode page (see 6.3.6), perform error recovery on the affected logical blocks;
- b) perform no automatic reallocation of the affected logical blocks, including any automatic reallocation enabled by the Read-Write Error Recovery mode page;
- c) consider errors on the affected logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see SPC-4);
- d) log errors on the affected logical blocks in the Error Counter log pages (see SPC-4);
- e) in any information returned for the error (e.g., in sense data or in the Background Scan Results log page (see 6.2.2)), set the sense key set to MEDIUM ERROR and either:
  - A) should set the additional sense code to READ ERROR – LBA MARKED BAD BY APPLICATION CLIENT; or
  - B) may set the additional sense code to UNRECOVERABLE READ ERROR.

#### 4.14.3 Block commands sense data descriptor

Table 6 defines the block commands sense data descriptor used in descriptor format sense data for direct-access block devices.

**Table 6 — Block commands sense data descriptor format**

Byte	Bit	7	6	5	4	3	2	1	0
0		DESCRIPTOR TYPE (05h)							
1		ADDITIONAL LENGTH (02h)							
2		Reserved							
3		Reserved		ILI	Reserved				

The DESCRIPTOR TYPE field and ADDITIONAL LENGTH field are defined in SPC-4 and shall be set to the value defined in table 6.

The INCORRECT LENGTH INDICATION (ILI) bit indicates that the requested data length in a READ LONG command or WRITE LONG command did not match the length of the logical block.

## 4.15 Model for XOR commands

### 4.15.1 Model for XOR commands overview

In storage arrays, a storage array controller (see 3.1.53) organizes a group of direct-access block devices into objects. The type of object supported by this model is the redundancy group (see 3.1.48), where some of the logical blocks on the direct-access block devices are used for XOR-protected data (see 3.1.61) and some of the logical blocks are used for check data (see 3.1.5). The check data is generated by performing a cumulative XOR (see 3.1.19) operation of the XOR-protected data. The XOR operation may be performed by the storage array controller or by the direct-access block devices.

A direct-access block device containing XOR-protected data is called a data disk. A direct-access block device containing check data is called a parity disk.

Performing the XOR operation in the direct-access block devices may result in a reduced number of data transfers across a service delivery subsystem. For example, when the XOR operation is done within the storage array controller, four commands are needed for a typical update write sequence:

- a) a command performing a read operation from the data disk;
- b) a command performing a write operation to the data disk;
- c) a command performing a read operation from the parity disk; and
- d) a command performing a write operation to the parity disk.

The storage array controller also does two internal XOR operations in this sequence.

In contrast, during storage array controller supervised XOR operations (see 4.15.2) only three commands are needed:

- a) a command performing a write operation to the data disk;
- b) a command performing a read operation from the data disk; and
- c) a command performing a write operation to the parity disk.

### 4.15.2 Storage array controller supervised XOR operations

#### 4.15.2.1 Storage array controller supervised XOR operations overview

A storage array controller supervises three basic operations that require XOR functionality:

- a) update write operation (see 4.15.2.2);
- b) regenerate operation (see 4.15.2.3); and
- c) rebuild operation (see 4.15.2.4).

Command sequences for each of these operations use the device servers in the direct-access block devices to perform the necessary XOR functions.

Three XOR commands are needed to implement storage array controller supervised XOR operations: XDREAD commands (see 5.40 and 5.41), XDWRITE commands (see 5.42 and 5.43), and XPWRITE commands (see 5.46 and 5.47). An XDWRITEREAD command (see 5.44 and 5.45) may be used in place of a sequence of an XDWRITE command followed by an XDREAD command. The storage array controller also uses READ commands and WRITE commands for certain operations.

#### 4.15.2.2 Update write operation

The update write operation writes new XOR-protected data to a data disk and updates the check data on the parity disk. The sequence is:

- 1) An XDWRITE command is sent to the data disk. This transfers new XOR-protected data to the data disk. The device server reads the old XOR-protected data, performs an XOR operation using the old XOR-protected data and the received XOR-protected data, retains the intermediate XOR result, and writes the received XOR-protected data to the medium;

- 2) An XDREAD command is sent to the data disk. This command transfers the intermediate XOR data to the storage array controller; and
- 3) An XPWRITE command is sent to the parity disk. This transfers the intermediate XOR data (i.e., XOR data received in a previous XDREAD command) to the parity disk. The device server reads the old check data, performs an XOR operation using the old check data and the intermediate XOR data, and writes the result (i.e., the new check data) to the medium.

In place of steps 1) and 2), a single XDWRITEREAD command may be sent.

#### 4.15.2.3 Regenerate operation

The regenerate operation is used to recreate one or more logical blocks that have an error. This is accomplished by reading the associated logical block from each of the other direct-access block devices within the redundancy group and performing an XOR operation with each of these logical blocks. The last XOR result is the data that should have been present on the unreadable direct-access block device. The number of steps is dependent on the number of direct-access block devices in the redundancy group, but the sequence is as follows:

- 1) A READ command is sent to the first direct-access block device. This transfers the data from the direct-access block device to the storage array controller;
- 2) An XDWRITE command with the DISABLE WRITE bit set to one is sent to the next direct-access block device, transferring the data from the previous read operation to the next direct-access block device. The direct-access block device reads its data, performs an XOR operation on the received data and its data, and retains the intermediate XOR result;
- 3) An XDREAD command is sent to the same direct-access block device as in step 2). This transfers the intermediate XOR data from the device to the storage array controller; and
- 4) Steps 2) and 3) are repeated until all direct-access block devices in the redundancy group except the failed device have been accessed.

The intermediate XOR data returned by the last XDREAD command is the regenerated data for the failed device.

In place of steps 2) and 3), a single XDWRITEREAD command with the DISABLE WRITE bit set to one may be used.

#### 4.15.2.4 Rebuild operation

The rebuild operation is similar to the regenerate operation, except that the last XOR result is written to the replacement device. This function is used when a failed device is replaced and the storage array controller is writing the rebuilt data to the replacement device. The number of steps is dependent on the number of direct-access block devices in the redundancy group, but the sequence is as follows:

- 1) A READ command is sent to the first direct-access block device. This transfers the data from the direct-access block device to the storage array controller;
- 2) An XDWRITE command with the DISABLE WRITE bit set to one is sent to the next direct-access block device, transferring the data from the previous read operation to the next direct-access block device. The device server reads its data, performs an XOR operation using the received data and its data, and retains the intermediate XOR result;
- 3) An XDREAD command is sent to the same direct-access block device as in step 2). This transfers the intermediate XOR data from the device to the storage array controller;
- 4) Steps 2) and 3) are repeated until all direct-access block devices in the redundancy group except the replacement device have been accessed. The intermediate XOR data returned by the last XDREAD command is the regenerated data for the replacement device; and
- 5) A WRITE command is sent to the replacement device. This transfers the regenerated data from step 4 to the replacement device. The replacement device writes the regenerated data to the medium.

In place of steps 2) and 3), a single XDWRITEREAD command with the DISABLE WRITE bit set to one may be used.

### 4.15.3 Array subsystem considerations

#### 4.15.3.1 Array subsystem considerations overview

This subclause lists considerations that apply to any array subsystem, but describes how use of the XOR commands may affect handling of those situations.

#### 4.15.3.2 Buffer full status handling

When the storage array controller sends an XDWRITE command to a device, the device retains the resulting XOR data until the storage array controller issues a matching XDREAD command to retrieve the data (see 4.15.4). Depending on the size of the device's buffer and the size of the XOR data, this may consume all of the device's internal buffer space. When all of the device's internal buffer space is allocated for XOR data, it may not be able to accept new medium access commands other than valid XDREAD commands and it may not be able to begin processing of commands that are already in the task set.

When the device server is not able to accept a new command because there is not enough space in the buffer, the device server shall terminate that command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to BUFFER FULL.

When a storage array controller receives this status, it may issue any matching XDREAD commands needed to satisfy any previous XDWRITE commands. This results in buffer space being freed for use by other commands. If it has no XDREAD commands to send, the storage array controller may assume the buffer space has been allocated to another SCSI initiator device (see SAM-4). The storage array controller may retry the command in the same manner that it would retry a command that a device server completes with TASK SET FULL status, including not retrying the command too frequently.

The bidirectional XDWRITEREAD command avoids the buffer full condition, since the device server controls when it accepts more write data and provides read data.

#### 4.15.3.3 Access to an inconsistent stripe

A stripe is a set of corresponding extents (see 3.1.20) from two or more direct-access block devices.

When the storage array controller issues an update write to a data disk, the data in the data disk has been updated when successful status is returned for the command. Until the parity disk has been updated, however, the associated stripe in the redundancy group is not consistent (i.e., performing an XOR operation on the XOR-protected data does not produce the check data).

The storage array controller shall keep track of this window of inconsistency and ensure that a regenerate or rebuild operation for any data extent within the stripe is not attempted until after the parity disk has been updated, making the stripe consistent again. For multi-initiator systems, tracking the updates may be more complex because each storage array controller needs to ensure that a second storage array controller is not writing to a stripe that the first storage array controller is regenerating or rebuilding. The coordination between storage array controllers is system specific and is beyond the scope of this standard.

If a device server terminates any of the XOR commands with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe may result. It is the storage array controller's responsibility to identify the failing device, the identify the scope of the failure, then limit access to the inconsistent stripe. The recovery procedures that the storage array controller implements are outside the scope of this standard.

#### 4.15.4 XOR data retention requirements

The device server shall retain XOR data resulting from an XDWRITE command awaiting retrieval by a matching XDREAD command until one of the following events occurs:

- a) a matching XDREAD command;
- b) logical unit reset;
- c) L\_T nexus loss associated with the L\_T nexus that sent the XDWRITE command;
- d) processing any of the following task management functions (see SAM-4):
  - A) CLEAR TASK SET;

- B) ABORT TASK specifying the I\_T\_L\_Q nexus (see SAM-4) of an XDREAD command retrieving that XOR data; or
- C) ABORT TASK SET.

If the XOR data is lost and the application client still wants to perform the XOR operation, it is required to resent the XDWRITE command after one of those events.

## 4.16 START STOP UNIT and power conditions

### 4.16.1 START STOP UNIT and power conditions overview

The START STOP UNIT command (see 5.19) allows an application client to control the power condition of a logical unit. This method includes specifying that the logical unit transition to a power condition.

In addition to the START STOP UNIT command, the power condition of a logical unit may be controlled by the Power Condition mode page (see SPC-4). If both the START STOP UNIT command and the Power Condition mode page methods are being used to control the power condition of the same logical unit, then the power condition specified by any START STOP UNIT command shall override the Power Condition mode page's power control.

There shall be no notification to the application client that a logical unit has transitioned from one power condition to another. The REQUEST SENSE command (see SPC-4) indicates if a logical unit is in the idle power condition or the standby power condition and may indicate if a logical unit is in the stopped power condition.

If the logical unit is in the idle power condition, then the device server shall process a REQUEST SENSE command by:

- 1) returning parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to:
  - A) LOW POWER CONDITION ON if the reason for entry into the idle power condition is unknown;
  - B) IDLE CONDITION ACTIVATED BY TIMER if the logical unit entered the idle power condition due to the idle condition timer (see SPC-4); and
  - C) IDLE CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle power condition due to a START STOP UNIT command or receipt of a command requiring the idle power condition while it was in the standby power condition;
 and
- 2) completing the REQUEST SENSE command with GOOD status.

If the logical unit is in the standby power condition, then the device server shall process a REQUEST SENSE command by:

- 1) returning parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to:
  - A) LOW POWER CONDITION ON if the reason for entry into the standby power condition is unknown;
  - B) STANDBY CONDITION ACTIVATED BY TIMER if the logical unit entered the standby power condition due to the standby condition timer (see SPC-4); and
  - C) STANDBY CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby power condition due to a START STOP UNIT command;
 and
- 2) completing the REQUEST SENSE command with GOOD status.

If the logical unit is in the stopped power condition, then the device server shall process a REQUEST SENSE command by:

- 1) returning parameter data containing sense data with:
  - A) the sense key set to NO SENSE and the additional sense code set to NO ADDITIONAL SENSE INFORMATION; or
  - B) the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED;
 and



- 2) completing the REQUEST SENSE command with GOOD status.

No power condition shall affect the supply of any power required for proper operation of a service delivery subsystem.

#### **4.16.2 START STOP UNIT and power conditions state machine**

##### **4.16.2.1 START STOP UNIT and power conditions state machine overview**

The SSU\_PC (start stop unit power condition) state machine for logical units implementing the START STOP UNIT command describes the logical unit power states and transitions resulting from settings by the START STOP UNIT command and settings in the Power Condition mode page (see SPC-4).

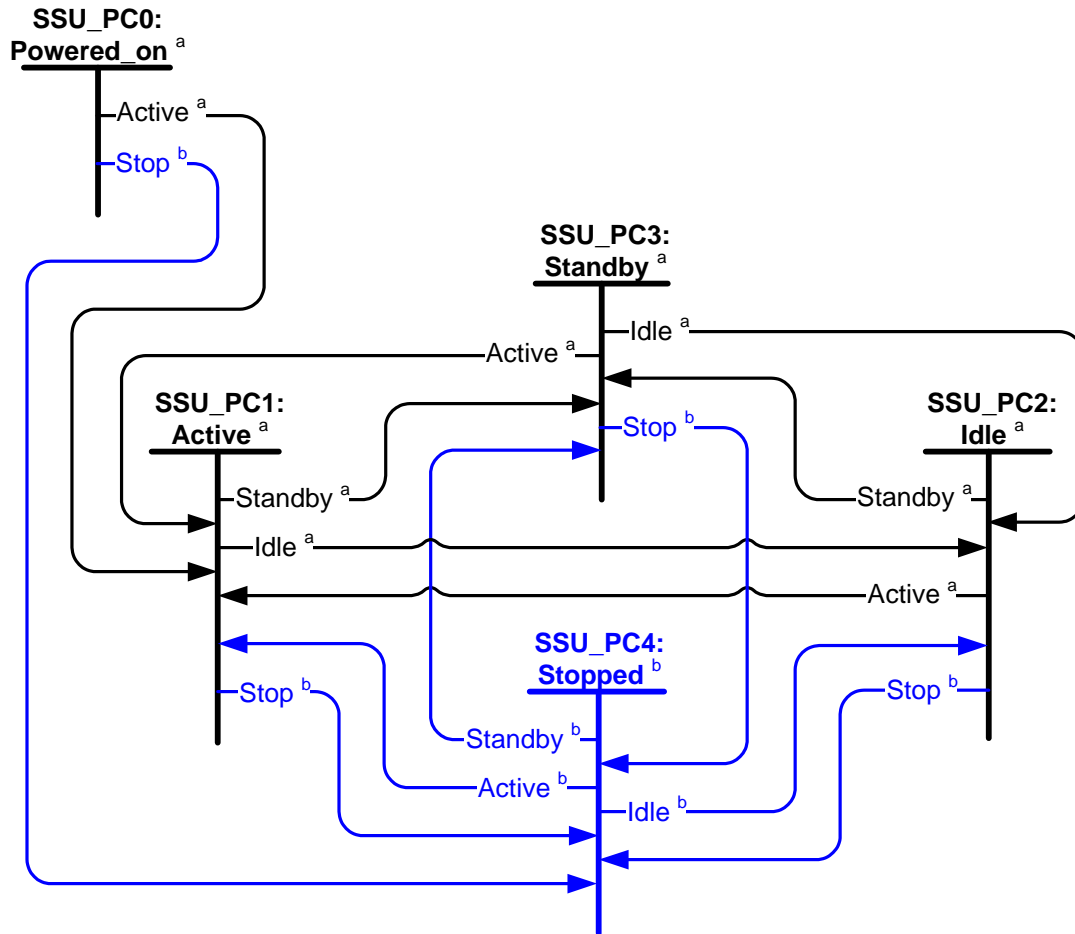
The SSU\_PC states are as follows:

- a) SSU\_PC0:Powered\_on (see 4.16.2.2) (initial state);
- b) SSU\_PC1:Active (see 4.16.2.3);
- c) SSU\_PC2:Idle (see 4.16.2.4);
- d) SSU\_PC3:Standby (see 4.16.2.5); and
- e) SSU\_PC4:Stopped (see 4.16.2.6).

The SSU\_PC state machine shall start in the SSU\_PC0:Powered\_on state after power on.

NOTE 6 - The SSU\_PC state machine is an enhanced version of the Power Condition state machine described in SPC-4.

Figure 4 describes the SSU\_PC state machine.



Notes:

<sup>a</sup> This state or transition is also described in SPC-4, but may have additional characteristics unique to this standard (e.g., a transition to or from a state described in this standard).

<sup>b</sup> This state or transition is described in this standard.

**Figure 4 — Power condition state machine for logical units implementing the START STOP UNIT command**

#### 4.16.2.2 SSU\_PC0:Powered\_on state

##### 4.16.2.2.1 SSU\_PC0:Powered\_on state description

The logical unit shall enter this state upon power on. This state consumes zero time.

##### 4.16.2.2.2 Transition SSU\_PC0:Powered\_on to SSU\_PC1:Active

This transition shall occur if:

- a) the logical unit has been configured to transition to the SSU\_PC1:Active state.

##### 4.16.2.2.3 Transition SSU\_PC0:Powered\_on to SSU\_PC4:Stopped

This transition shall occur if:

- a) the logical unit has been configured to transition to the SSU\_PC4:Stopped state.

#### **4.16.2.3 SSU\_PC1:Active state**

##### **4.16.2.3.1 SSU\_PC1:Active state description**

While in this state, if power on initialization is not complete, then the logical unit completes its power on initialization.

While in this state, after power on initialization is complete, then:

- a) the logical unit is in the active power condition (see SPC-4);
- b) if the idle condition timer is active (see SPC-4) and not disabled (see 5.19), then the idle condition timer is running; and
- c) if the standby condition timer is active (see SPC-4) and not disabled (see 5.19), then the standby condition timer is running.

##### **4.16.2.3.2 Transition SSU\_PC1:Active to SSU\_PC2:Idle**

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to IDLE;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_IDLE\_0; or
- c) the idle condition timer is active (see SPC-4), enabled (see 5.19), and zero.

##### **4.16.2.3.3 Transition SSU\_PC1:Active to SSU\_PC3:Standby**

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to STANDBY;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_STANDBY\_0; or
- c) the standby condition timer is active (see SPC-4), enabled (see 5.19), and zero.

##### **4.16.2.3.4 Transition SSU\_PC1:Active to SSU\_PC4:Stopped**

This transition shall occur after the device server processes a START STOP UNIT command with the START bit set to zero and the POWER CONDITION field set to START\_VALID.

#### **4.16.2.4 SSU\_PC2:Idle state**

##### **4.16.2.4.1 SSU\_PC2:Idle state description**

While in this state:

- a) the logical unit is in the idle power condition (see SPC-4);
- b) the device server processes the REQUEST SENSE command as described in 4.16.1; and
- c) if the standby condition timer is active (see SPC-4) and not disabled (see 5.19), then the standby condition timer is running.

##### **4.16.2.4.2 Transition SSU\_PC2:Idle to SSU\_PC1:Active**

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the START bit set to one;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to ACTIVE; or
- c) the device server processes a command that requires the logical unit to be in the SSU\_PC1:Active state to process the command.

#### 4.16.2.4.3 Transition SSU\_PC2:Idle to SSU\_PC3:Standby

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to STANDBY;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_STANDBY\_0; or
- c) the standby condition timer is active (see SPC-4), enabled (see 5.19), and zero.

#### 4.16.2.4.4 Transition SSU\_PC2:Idle to SSU\_PC4:Stopped

This transition shall occur after the device server processes a START STOP UNIT command with the START bit set to zero.

#### 4.16.2.5 SSU\_PC3:Standby state

##### 4.16.2.5.1 SSU\_PC3:Standby state description

While in this state:

- a) the logical unit is in the standby power condition (see SPC-4); and
- b) the device server processes the REQUEST SENSE command as described in 4.16.1.

##### 4.16.2.5.2 Transition SSU\_PC3:Standby to SSU\_PC1:Active

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the START bit set to one;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to ACTIVE; or
- c) the device server processes a command that requires the logical unit to be in the SSU\_PC1:Active state to process the command.

##### 4.16.2.5.3 Transition SSU\_PC3:Standby to SSU\_PC2:Idle

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to IDLE;
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_IDLE\_0; or
- c) the device server processes a command that requires the logical unit to be in the SSU\_PC2:Idle state to process the command.

##### 4.16.2.5.4 Transition SSU\_PC3:Standby to SSU\_PC4:Stopped

This transition shall occur after the device server processes a START STOP UNIT command with the START bit set to zero.

#### 4.16.2.6 SSU\_PC4:Stopped state

##### 4.16.2.6.1 SSU\_PC4:Stopped state description

While in this state:

- a) the logical unit is in the stopped power condition;
- b) the device server is not capable of processing medium access commands. While in this state, the device server shall terminate each medium access command or TEST UNIT READY command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED;
- c) the device server processes the REQUEST SENSE command as described in 4.16.1; and

- d) the power consumed by the SCSI target device should be less than or equal to that consumed than when the logical unit is in the SSU\_PC1:Active, SSU\_PC2:Idle, or SSU\_PC3:Standby states.

#### 4.16.2.6.2 Transition SSU\_PC4:Stopped to SSU\_PC1:Active

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the START bit set to one; or
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to ACTIVE.

#### 4.16.2.6.3 Transition SSU\_PC4:Stopped to SSU\_PC2:Idle

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to IDLE; or
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_IDLE\_0.

#### 4.16.2.6.4 Transition SSU\_PC4:Stopped to SSU\_PC3:Standby

This transition shall occur after:

- a) the device server processes a START STOP UNIT command with the POWER CONDITION field set to STANDBY; or
- b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to FORCE\_STANDBY\_0.

### 4.17 Protection information model

#### 4.17.1 Protection information overview

The protection information model provides for protection of user data while it is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object associated with the I\_T\_L nexus (see SAM-4). Once received, protection information is retained (e.g., written to medium, stored in non-volatile memory, or recalculated on read back) by the device server until overwritten. Power loss, hard reset, logical unit reset, and I\_T nexus loss shall have no effect on the retention of protection information.

Support for protection information shall be indicated in the PROTECT bit in the standard INQUIRY data (see SPC-4).

If the logical unit is formatted with protection information and the EMDP bit is set to one in the Disconnect-Reconnect mode page (see SPC-4), then checking of the logical block reference tag within a service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

Protection information is also referred to as the data integrity field (DIF).

#### 4.17.2 Protection types

##### 4.17.2.1 Protection types overview

The content of protection information is dependent on the type of protection to which a logical unit has been formatted.

The type of protection supported by the logical unit shall be indicated in the SPT field in the Extended INQUIRY Data VPD page (see SPC-4). The current protection type shall be indicated in the P\_TYPE field in the READ CAPACITY(16) command (see 5.13).

An application client may format the logical unit to a specific type of protection using the FMTPINFO field and the PROTECTION FIELD USAGE field in the FORMAT UNIT command (see 5.2).

The medium access commands are processed in a different manner by a device server depending on the type of protection in effect. When used in relation to types of protection, the term “medium access commands” is defined as the following commands:

- a) ORWRITE;
- b) READ (10);
- c) READ (12);
- d) READ (16);
- e) READ (32);
- f) VERIFY (10);
- g) VERIFY (12);
- h) VERIFY (16);
- i) VERIFY (32);
- j) WRITE (10);
- k) WRITE (12);
- l) WRITE (16);
- m) WRITE (32);
- n) WRITE AND VERIFY (10);
- o) WRITE AND VERIFY (12);
- p) WRITE AND VERIFY (16);
- q) WRITE AND VERIFY (32);
- r) WRITE SAME (10);
- s) WRITE SAME (16);
- t) WRITE SAME (32);
- u) XDWRITE (10);
- v) XDWRITE (32);
- w) XDWRITEREAD (10);
- x) XDWRITEREAD (32);
- y) XPWRITE (10);
- z) XPWRITE (32);
- aa) XDREAD (10); and
- ab) XDREAD (32).

The device server may allow the READ (6) command (see 5.7) and the WRITE (6) command (see 5.26) regardless of the type of protection to which the logical unit has been formatted.

#### 4.17.2.2 Type 0 protection

Type 0 protection defines no protection over that which is defined within the transport protocol.

A logical unit that has been formatted with protection information disabled (see 5.2) or a logical unit that does not support protection information (i.e., the PROTECT bit set to zero in the Standard INQUIRY data (see SPC-4)) has type 0 protection.

If type 0 protection is enabled and the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to a non-zero value, then media commands are invalid and may be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If type 0 protection is enabled and the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to a zero value, then the following media commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

- a) READ (32);
- b) VERIFY (32);
- c) WRITE (32);
- d) WRITE AND VERIFY (32); and
- e) WRITE SAME (32).

#### 4.17.2.3 Type 1 protection

Type 1 protection:

- a) defines the content of the LOGICAL BLOCK GUARD field;
- b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
- c) defines the content the LOGICAL BLOCK REFERENCE TAG field.

If type 1 protection is enabled, then the following media commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

- a) READ (32);
- b) VERIFY (32);
- c) WRITE (32);
- d) WRITE AND VERIFY (32); and
- e) WRITE SAME (32).

For valid medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to:

- a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or
- b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

#### 4.17.2.4 Type 2 protection

Type 2 protection:

- a) defines the content of the LOGICAL BLOCK GUARD field;
- b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
- c) defines, except for the first logical block addressed by the command, the content of the LOGICAL BLOCK REFERENCE TAG field.

If type 2 protection is enabled and the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to a non-zero value, then the following media commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

- a) ORWRITE;
- b) READ (10);
- c) READ (12);
- d) READ (16);
- e) VERIFY (10);
- f) VERIFY (12);
- g) VERIFY (16);
- h) WRITE (10);
- i) WRITE (12);
- j) WRITE (16);
- k) WRITE AND VERIFY (10);
- l) WRITE AND VERIFY (12);
- m) WRITE AND VERIFY (16);
- n) WRITE SAME (10);
- o) WRITE SAME (16);
- p) XDWRITE (10);
- q) XDWRITE (32);
- r) XDWRITEREAD (10);
- s) XDWRITEREAD (32);
- t) XPWRITE (10);
- u) XPWRITE (32);
- v) XDREAD (10); and

w) XDREAD (32).

For valid medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to:

- a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or
- b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

#### 4.17.2.5 Type 3 protection

Type 3 protection:

- a) defines the content of the LOGICAL BLOCK GUARD field within the logical blocks of the data-in buffer and/or data-out buffer;
- b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
- c) does not define the content of the LOGICAL BLOCK REFERENCE TAG field.

If type 3 protection is enabled, then the following media commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

- a) READ (32);
- b) VERIFY (32);
- c) WRITE (32);
- d) WRITE AND VERIFY (32); and
- e) WRITE SAME (32).

For valid medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to:

- a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or
- b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

#### 4.17.3 Protection information format

Table 7 defines the placement of protection information in a logical block.

**Table 7 — User data and protection information format**

Bit	7	6	5	4	3	2	1	0
Byte								
0	USER DATA							
n - 1								
n	(MSB)	LOGICAL BLOCK GUARD						
n + 1								(LSB)
n + 2	(MSB)	LOGICAL BLOCK APPLICATION TAG						
n + 3								(LSB)
n + 4	(MSB)	LOGICAL BLOCK REFERENCE TAG						
n + 7								(LSB)



The USER DATA field shall contain user data. The contents of the USER DATA field shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK GUARD field contains the CRC (see 4.17.4) of the contents of the USER DATA field.

The LOGICAL BLOCK APPLICATION TAG field is set by the application client. If the device server detects a:

- a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or
- b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF\_FFFFh, and type 3 protection (see 4.17.2.5) is enabled,

then the device server disables checking of all protection information for the logical block when reading from the medium. Otherwise, the contents of the logical block application tag are not defined by this standard.

The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field.

The contents of the LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the value specified in table 8.

**Table 8 — Contents of the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer**

Protection Type	Content of the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer
Type 1 protection (see 4.17.2.3)	The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command.
Type 2 protection (see 4.17.2.4)	The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command.
Type 3 protection (see 4.17.2.5)	Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field.

The LOGICAL BLOCK REFERENCE TAG field subsequent logical blocks in the data-in buffer and/or data-out buffer shall be set as specified in table 9.

**Table 9 — Setting the LOGICAL BLOCK REFERENCE TAG field of the subsequent logical blocks in the data-in buffer and/or data-out buffer**

Protection Type	The content of the LOGICAL BLOCK REFERENCE TAG field of each subsequent logical block in the data-in buffer and/or data-out buffer
Type 1 protection (see 4.17.2.3) and Type 2 protection (see 4.17.2.4)	The logical block reference tag of the previous logical block plus one.
Type 3 protection (see 4.17.2.5)	Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field.

The contents of the LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

#### 4.17.4 Logical block guard

##### 4.17.4.1 Logical block guard overview

The LOGICAL BLOCK GUARD field shall contain a CRC that is generated from the contents of the USER DATA field.

Table 10 defines the CRC polynomials used to generate the logical block guard from the contents of the USER DATA field.

**Table 10 — CRC polynomials**

Function	Definition
$F(x)$	A polynomial representing the transmitted USER DATA field, which is covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be byte zero bit seven of the USER DATA field and the coefficient of the lowest order term shall be bit zero of the last byte of the USER DATA field.
$F'(x)$	A polynomial representing the received USER DATA field.
$G(x)$	The generator polynomial: $G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (i.e., $G(x) = 18BB7h$ )
$R(x)$	The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted LOGICAL BLOCK GUARD field.
$R'(x)$	A polynomial representing the received LOGICAL BLOCK GUARD field.
$RB(x)$	The remainder polynomial calculated during CRC checking by the receiver. $RB(x) = 0$ indicates no error was detected.
$RC(x)$	The remainder polynomial calculated during CRC checking by the receiver. $RC(x) = 0$ indicates no error was detected.
$QA(x)$	The quotient polynomial calculated during CRC generation by the transmitter. The value of $QA(x)$ is not used.
$QB(x)$	The quotient polynomial calculated during CRC checking by the receiver. The value of $QB(x)$ is not used.
$QC(x)$	The quotient polynomial calculated during CRC checking by the receiver. The value of $QC(x)$ is not used.
$M(x)$	A polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field.
$M'(x)$	A polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field.

##### 4.17.4.2 CRC generation

The equations that are used to generate the CRC from  $F(x)$  are as follows. All arithmetic is modulo 2.

The transmitter shall calculate the CRC by appending 16 zeros to  $F(x)$  and dividing by  $G(x)$  to obtain the remainder  $R(x)$ :

$$\frac{(x^{16} \times F(x))}{G(x)} = QA(x) + \frac{R(x)}{G(x)}$$

$R(x)$  is the CRC value, and is transmitted in the LOGICAL BLOCK GUARD field.

$M(x)$  is the polynomial representing the USER DATA field followed by the LOGICAL BLOCK GUARD field (i.e.,  $F(x)$  followed by  $R(x)$ ):

$$M(x) = (x^{16} \times F(x)) + R(x)$$

#### 4.17.4.3 CRC checking

$M'(x)$  (i.e., the polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field) may differ from  $M(x)$  (i.e., the polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field) if there are transmission errors.

The receiver may check  $M'(x)$  validity by appending 16 zeros to  $F'(x)$  and dividing by  $G(x)$  and comparing the calculated remainder  $RB(x)$  to the received CRC value  $R'(x)$ :

$$\frac{(x^{16} \times F'(x))}{G(x)} = QB(x) + \frac{RB(x)}{G(x)}$$

In the absence of errors in  $F'(x)$  and  $R'(x)$ , the remainder  $RB(x)$  is equal to  $R'(x)$ .

The receiver may check  $M'(x)$  validity by dividing  $M'(x)$  by  $G(x)$  and comparing the calculated remainder  $RC(x)$  to zero:

$$\frac{M'(x)}{G(x)} = QC(x) + \frac{RC(x)}{G(x)}$$

In the absence of errors in  $F'(x)$  and  $R'(x)$ , the remainder  $RC(x)$  is equal to zero.

Both methods of checking  $M'(x)$  validity are mathematically equivalent.

#### 4.17.4.4 CRC test cases

Several CRC test cases are shown in table 11.

**Table 11 — CRC test cases**

Pattern	CRC
32 bytes each set to 00h	0000h
32 bytes each set to FFh	A293h
32 bytes of an incrementing pattern from 00h to 1Fh	0224h
2 bytes each set to FFh followed by 30 bytes set to 00h	21B8h
32 bytes of a decrementing pattern from FFh to E0h	A0B7h

#### 4.17.5 Application of protection information

Before an application client transmits or receives logical blocks with protection information it shall:

- 1) determine if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-4);
- 2) if protection information is supported, then determine if the logical unit is formatted to accept protection information using the READ CAPACITY (16) command (see the PROT\_EN bit in 5.13); and
- 3) if the logical unit supports protection information and is not formatted to accept protection information, then format the logical unit with protection information enabled.

If the logical unit supports protection information and is formatted to accept protection information, then the application client may use commands performing read operations that support protection information and should use commands performing write and verify operations that support protection information.

#### 4.17.6 Protection information and commands

The enabling of protection information enables fields in some commands that instruct the device server on the handling of protection information. The detailed definitions of each command's protection information fields are in the individual command descriptions.

The commands that are affected when protection information is enabled are listed in table 13 (see 5.1).

Commands that cause a device server to return the length in bytes of each logical block (e.g., the MODE SENSE commands and the READ CAPACITY commands) shall cause the device server to return the length of the USER DATA field, not including the length of the protection information (i.e., the LOGICAL BLOCK GUARD field, the LOGICAL BLOCK APPLICATION TAG field, and the LOGICAL BLOCK REFERENCE TAG field) (e.g., if the user data plus the protection information is equal to 520 bytes then 512 is returned).

#### 4.18 Grouping function

A grouping function is a function that collects information about attributes associated with commands (i.e., information about commands with the same group value are collected into the specified group). The definition of the attributes and the groups is outside the scope of this standard. Groups are identified with the GROUP NUMBER field in the CDB of certain commands (e.g., the PRE-FETCH (10) command (see 5.4)).

The collection of this information is outside the scope of this standard (e.g., the information may not be transmitted using any SCSI protocols).

NOTE 7 - An example of how grouping could be used, consider two applications using a subsystem; one application streams data and another accesses data randomly. If the streaming application groups all of its commands with one value (e.g., x), and the random application groups all of its commands with another value (e.g., y), then a group x defined to hold performance metrics collects all the performance metrics for the streamed commands together and a group y defined to also hold performance metrics collects all the performance metrics for the random commands together. The result is two sets of performance metrics (i.e., x and y). A management application then reads the performance metrics and determines if the performance of a specific group is acceptable.

Support for the grouping function is indicated in the GROUP\_SUP bit in the Extended INQUIRY Data VPD page (see SPC-4).

#### 4.19 Background scanning operations

##### 4.19.1 Background scanning overview

During background scanning, the device server, without using any bandwidth on a service delivery subsystem, reads logical blocks from the medium for the purpose of:

- a) identifying logical blocks that are difficult to read or unreadable;
- b) logging read problems; and
- c) when allowed, taking a vendor-specific action to make the logical block readable again.

If a logical block is readable but requires extra actions (e.g., retries or application of a correction algorithm) to be read (i.e., the data is recoverable), then the device server may resolve the problem using vendor-specific means. The ARRE bit in the Read-Write Error Recovery mode page (see 6.3.5) controls whether the device server performs automatic reallocation of the defective logical blocks during read operations.

If a logical block is unreadable (i.e., the data is unrecoverable), then the device server may mark the logical block as bad so it may be relocated. The AWRE bit in the Read-Write Error Recovery mode page (see 6.3.5) controls whether the device server performs automatic reallocation of the defective logical blocks during write operations. If allowed by the AWRE bit setting, logical blocks that have previously been noted as unrecoverable are reassigned at the start of the next write operation to that logical block.

During a background scan, the device server may scan the logical blocks in any order (e.g., based on physical block layout). The device server should not retain any logical blocks in cache memory after they are read.

During a background scan, the device server shall ignore pseudo unrecoverable errors with correction disabled, and shall process pseudo unrecoverable errors with correction enabled.

## 4.19.2 Background pre-scan

### 4.19.2.1 Enabling the background pre-scan operation

The background pre-scan operation is enabled after:

- 1) the EN\_PS bit in the Background Control mode page (see 6.3.3) is set to zero;
- 2) the EN\_PS bit is set to one; and
- 3) the SCSI device is power cycled if;
  - A) the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to zero, or if the S\_L\_FULL bit is set to one and the log is not full; and
  - B) the saved value of the EN\_PS bit is set to one.

After the background pre-scan operation is enabled, the device server shall:

- a) initialize the Background Pre-scan Time Limit timer to the time specified in the PRE-SCAN TIME LIMIT field in the Background Control mode page and start the timer;
- b) initialize the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field in the Background Control mode page and start the timer; and
- c) begin the background pre-scan operation (i.e., begin scanning the medium).

### 4.19.2.2 Suspending the background pre-scan operation

The device server shall suspend the background pre-scan operation when any of the following occurs:

- a) a command is received from the application client; or
- b) the Background Scanning Results log page Background Medium Scan log parameters are all used and the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to one.

When a command is received from the application client during the background pre-scan operation, the background pre-scan operation should be suspended within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field in the Background Control mode page (see 6.3.3).

While the background pre-scan operation is suspended and not halted (see 4.19.3.2), the device server shall convert each write operation that accesses a logical unit that has not been scanned during the background pre-scan operation into a write operation followed by a verify operation to verify that the data just written was read back successfully. If a write operation accesses a logical unit that has already been scanned during the background pre-scan operation then the write operation shall be processed normally. Commands that do not perform write operations shall be processed normally.

The suspended background pre-scan operation shall resume where it left off when:

- a) all commands have been completed;
- b) no ACA condition exists;
- c) the Background Scanning Results log page Background Medium Scan log parameters are not all used or the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to zero;
- d) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page (see 6.3.3); and
- e) the background pre-scan operation has not been halted (see 4.19.3.2).

### 4.19.2.3 Halting the background pre-scan operation

The device server shall halt the background pre-scan operation if any of the following occurs:

- a) the background pre-scan operation completes scanning all logical blocks on the medium;
- b) an application client sets the EN\_PS bit to zero (see 6.3.3);
- c) the Background Pre-scan Time Limit timer expires;
- d) the device server detects a fatal error;
- e) the device server detects a vendor-specific pattern of errors;
- f) the device server detects a medium formatted without a PLIST (see 4.9); or
- g) the device server detects temperature out of range.

To re-enable background pre-scan operation after it is halted, use the procedure described in 4.19.2.1.

### 4.19.3 Background medium scan

#### 4.19.3.1 Enabling the background medium scan operation

The background medium scan operation is enabled if:

- a) the background pre-scan operation (see 4.19.2) is not enabled;
- b) the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to zero or the S\_L\_FULL bit is set to one and the log is not full; and
- c) the EN\_BMS bit in the Background Control mode page (see 6.3.3) is set to one.

If the background medium scan operation is enabled, the device server shall begin the background medium scan operation (i.e., begin scanning the medium) when:

- a) the Background Medium Scan Interval timer has expired; and
- b) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page (see 6.3.3).

After power on, if the background pre-scan operation is not enabled (see 4.19.2.1), the device server shall set the Background Medium Scan Interval timer to zero (i.e., expired).

Whenever the background medium scan operation begins, the device server shall set the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field in the Background Control mode page and start the timer.

#### 4.19.3.2 Suspending the background medium scan operation

The device server shall suspend the background medium scan operation if any of the following occurs:

- a) a command is received from the application client;
- b) the Background Scanning Results log page Background Medium Scan log parameters are all used and the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to one;
- c) the background medium scan operation completes scanning all logical blocks on the medium; or
- d) an application client sets the EN\_BMS bit to zero (see 6.3.3).

When a command is received from the application client, the device server should suspend the background medium scan operation within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field in the Background Control mode page (see 6.3.3).

The suspended background medium scan operation shall resume where it left off when:

- a) all commands have been successfully completed;
- b) no ACA condition exists;
- c) the Background Scanning Results log page Background Medium Scan log parameters are not all used or the S\_L\_FULL bit in the Background Control mode page (see 6.3.3) is set to zero;
- d) the EN\_BMS bit is set to one; and
- e) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page (see 6.3.3).

#### 4.19.3.3 Halting the background medium scan operation

The device server shall halt the background medium scan operation if any of the following occurs:

- a) the background medium scan operation completes scanning all logical blocks on the medium;
- b) the device server detects a fatal error;
- c) the device server detects a vendor-specific pattern of errors;
- d) the device server detects a medium formatted without a PLIST (see 4.9); or
- e) the device server detects temperature out of range.

To re-enable background medium scan operation after it is halted, use the procedure described in 4.19.3.1.

#### 4.19.4 Interpreting the logged results

An application client may:

- a) poll the Background Scan Results log page (see 6.2.2) to get information about pre-scan and background medium scan activity; or
- b) use the EBACKERR bit and the MRIE field in the Informational Exceptions Control mode page (see SPC-4) to select a method of indicating that a medium error was detected. If the EBACKERR bit is set to one and a medium error was detected, then the device server shall return the following additional sense codes using the method defined in the MRIE field:
  - A) WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR if the failure occurs during a background pre-scan; or
  - B) WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR if the failure occurs during a background medium scan.

The Background Scan Results log page (see 6.2.2) Background Scanning Status parameter (see table 106) indicates whether a background pre-scan or background medium scan is active or suspended, the number of background scans performed on the medium, and the progress of a background scan that is active. This information may be used by an application client to monitor the background scanning operations and should be used by an application client after notification via an informational exception.

The Background Medium Scan parameters (see table 109), if any, describe the location of any suspected bad logical blocks. The REASSIGN STATUS field (see table 111) indicates whether the defect was completely handled by the device server or whether the application client is required to take action (e.g., reassigning or re-writing a logical block) to fix a particular bad logical block.

After an application client analyzes the Background Medium Scan parameters and has completed actions, if any, to repair the bad logical blocks, it may delete the log entries by issuing a LOG SELECT command (e.g., with the PCR bit set to one, or with the PC field set to 11b and the PARAMETER LIST LENGTH field set to zero) (see SPC-4).

The background medium scan continues to run during log page accesses. To ensure that the log page does not change during a sequence of accesses, the application client shall:

- 1) set the EN\_BMS bit in the Background Control mode page (see 6.3.3) to zero to suspend the background medium scan;
- 2) read the log page with LOG SENSE command;
- 3) process the log page;
- 4) delete the log entries with the LOG SELECT command (e.g., with the PCR bit set to one); and
- 5) set the EN\_BMS bit in the Background Control mode page (see 6.3.3) to one.

#### 4.20 Association between commands and CbCS permission bits

Table 12 defines the Capability based Command Security (i.e., CbCS) permissions required for each command defined in this standard to be processed by a secure CDB processor. The permissions shown in table 12 are defined in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4). This standard does not define any permission specific to block commands.

**Table 12 — Associations between commands and CbCS permissions (part 1 of 3)**

Command name	Permissions							
	DATA READ	DATA WRITE	PARAM READ	PARAM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
FORMAT UNIT		v		v				
ORWRITE		v						
PRE-FETCH (10)	v							
PRE-FETCH (16)	v							
<b>Key:</b> v = a secure CDB processor shall process the requested command only if the corresponding bit is set in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4).								

Table 12 — Associations between commands and CbCS permissions (part 2 of 3)

Command name	Permissions							
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
PREVENT ALLOW MEDIUM REMOVAL								v
READ (6)	v							
READ (10)	v							
READ (12)	v							
READ (16)	v							
READ (32)	v							
READ CAPACITY (10)			v					
READ CAPACITY (16)			v					
READ DEFECT DATA (10)			v					
READ DEFECT DATA (12)			v					
READ LONG (10)	v							
READ LONG (16)	v							
REASSIGN BLOCKS								
START STOP UNIT								v
SYNCHRONIZE CACHE (10)		v						
SYNCHRONIZE CACHE (16)		v						
VERIFY (10)	v							
VERIFY (12)	v							
VERIFY (16)	v							
VERIFY (32)	v							
WRITE (6)		v						
WRITE (10)		v						
WRITE (12)		v						
WRITE (16)		v						
WRITE (32)		v						
WRITE AND VERIFY (10)		v						
WRITE AND VERIFY (12)		v						
WRITE AND VERIFY (16)		v						
WRITE AND VERIFY (32)		v						
WRITE LONG (10)		v						
WRITE LONG (16)		v						
WRITE SAME (10)		v						
WRITE SAME (16)		v						
WRITE SAME (32)		v						
XDREAD (10)	v							
XDREAD (32)	v							
XDWRITE (10)		v						
XDWRITE (32)		v						
<b>Key:</b> v = a secure CDB processor shall process the requested command only if the corresponding bit is set in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4).								



**Table 12 — Associations between commands and CbCS permissions** (part 3 of 3)

Command name	Permissions							
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
XDWRITEREAD (10)	v	v						
XDWRITEREAD (32)	v	v						
XPWRITE (10)		v						
XPWRITE (32)		v						
<b>Key:</b> v = a secure CDB processor shall process the requested command only if the corresponding bit is set in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4).								

#### 4.21 Deferred microcode activation

After receiving a FORMAT UNIT command (see 5.2) or a START STOP UNIT command (see 5.19), a device server shall activate any deferred microcode that has been downloaded (see SPC-4) prior to processing the command.

## 5 Commands for direct-access block devices

### 5.1 Commands for direct-access block devices overview

The commands for direct-access block devices are listed in table 13. Commands with CDB or parameter data fields that support protection information (see 4.17) or for which protection information may be a factor in the processing of the command are indicated by the fourth (i.e., Protection information) column.

**Table 13 — Commands for direct-access block devices** (part 1 of 3)

Command name	Operation code <sup>a</sup>	Type <sup>b</sup>	Protection information	Reference
ACCESS CONTROL IN	86h	O	no	SPC-4
ACCESS CONTROL OUT	87h	O	no	SPC-4
CHANGE ALIASES	A4h/0Bh	O	no	SPC-4
EXTENDED COPY	83h	O	no	SPC-4
FORMAT UNIT	04h	M	yes	5.2
INQUIRY	12h	M	yes	SPC-4
LOG SELECT	4Ch	O	no	SPC-4
LOG SENSE	4Dh	O	no	SPC-4
MAINTENANCE IN	A3h/00h to 04h A3h/06h to 09h	X <sup>e</sup>	no	SCC-2
MAINTENANCE OUT	A4h/00h to 05h A4h/07h to 09h	X <sup>e</sup>	no	SCC-2
MODE SELECT (6)	15h	O	no	SPC-4
MODE SELECT (10)	55h	O	no	SPC-4
MODE SENSE (6)	1Ah	O	no	SPC-4
MODE SENSE (10)	5Ah	O	no	SPC-4
ORWRITE	8Bh	O	yes	5.3
PERSISTENT RESERVE IN	5Eh	O	no	SPC-4
PERSISTENT RESERVE OUT	5Fh	O	no	SPC-4
PRE-FETCH (10)	34h	O	no	5.4
PRE-FETCH (16)	90h	O	no	5.5
PREVENT ALLOW MEDIUM REMOVAL	1Eh	O	no	5.6
READ (6)	08h	M <sup>c</sup>	yes	5.7
READ (10)	28h	M	yes	5.8
READ (12)	A8h	O	yes	5.9
READ (16)	88h	O	yes	5.10
READ (32)	7Fh/0009h	O	yes	5.11
READ ATTRIBUTE	8Ch	O	no	SPC-4
READ BUFFER	3Ch	O	no	SPC-4
READ CAPACITY (10)	25h	M	no	5.12
READ CAPACITY (16)	9Eh/10h	X <sup>d</sup>	yes	5.13
READ DEFECT DATA (10)	37h	O	no	5.14
READ DEFECT DATA (12)	B7h	O	no	5.15
READ LONG (10)	3Eh	O	yes	5.16
READ LONG (16)	9Eh/11h	O	yes	5.17
REASSIGN BLOCKS	07h	O	no	5.18

Table 13 — Commands for direct-access block devices (part 2 of 3)

Command name	Operation code <sup>a</sup>	Type <sup>b</sup>	Protection information	Reference
RECEIVE COPY RESULTS	84h	O	no	SPC-4
RECEIVE DIAGNOSTIC RESULTS	1Ch	O/M <sup>f</sup>	no	SPC-4
REDUNDANCY GROUP IN	BAh	X <sup>e</sup>	no	SCC-2
REDUNDANCY GROUP OUT	BBh	X <sup>e</sup>	no	SCC-2
REPORT ALIASES	A3h/0Bh	O	no	SPC-4
REPORT DEVICE IDENTIFIER	A3h/05h	O	no	SPC-4
REPORT LUNS	A0h	M	no	SPC-4
REPORT PRIORITY	A3h/0Eh	O	no	SPC-4
REPORT SUPPORTED OPERATION CODES	A3h/0Ch	O	no	SPC-4
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh	O	no	SPC-4
REPORT TARGET PORT GROUPS	A3h/0Ah	O	no	SPC-4
REQUEST SENSE	03h	M	no	SPC-4
SECURITY PROTOCOL IN	A2h	O	no	SPC-4
SECURITY PROTOCOL OUT	B5h	O	no	SPC-4
SEND DIAGNOSTIC	1Dh	M	no	SPC-4
SET DEVICE IDENTIFIER	A4h/06h	O	no	SPC-4
SET PRIORITY	A4h/0Eh	O	no	SPC-4
SET TARGET PORT GROUPS	A4h/0Ah	O	no	SPC-4
SPARE IN	BCh	X <sup>e</sup>	no	SCC-2
SPARE OUT	BDh	X <sup>e</sup>	no	SCC-2
START STOP UNIT	1Bh	O	no	5.19
SYNCHRONIZE CACHE (10)	35h	O	no	5.20
SYNCHRONIZE CACHE (16)	91h	O	no	5.21
TEST UNIT READY	00h	M	no	SPC-4
VERIFY (10)	2Fh	O	yes	5.22
VERIFY (12)	AFh	O	yes	5.23
VERIFY (16)	8Fh	O	yes	5.24
VERIFY (32)	7Fh/000Ah	O	yes	5.25
VOLUME SET IN	BEh	X <sup>e</sup>	no	SCC-2
VOLUME SET OUT	BFh	X <sup>e</sup>	no	SCC-2
WRITE (6)	0Ah	O <sup>c</sup>	yes	5.26
WRITE (10)	2Ah	O	yes	5.27
WRITE (12)	AAh	O	yes	5.28
WRITE (16)	8Ah	O	yes	5.29
WRITE (32)	7Fh/000Bh	O	yes	5.30
WRITE AND VERIFY (10)	2Eh	O	yes	5.31
WRITE AND VERIFY (12)	A Eh	O	yes	5.32
WRITE AND VERIFY (16)	8Eh	O	yes	5.33
WRITE AND VERIFY (32)	7Fh/000Ch	O	yes	5.34
WRITE ATTRIBUTE	8Dh	O	no	SPC-4
WRITE BUFFER	3Bh	O	no	SPC-4

**Table 13 — Commands for direct-access block devices** (part 3 of 3)

Command name	Operation code <sup>a</sup>	Type <sup>b</sup>	Protection information	Reference
WRITE LONG (10)	3Fh	O	yes	5.35
WRITE LONG (16)	9Fh/11h	O	yes	5.36
WRITE SAME (10)	41h	O	yes	5.37
WRITE SAME (16)	93h	O	yes	5.38
WRITE SAME (32)	7Fh/000Dh	O	yes	5.39
XDREAD (10)	52h	O	yes	5.40
XDREAD (32)	7Fh/0003h	O	yes	5.41
XDWRITE (10)	50h	O	yes	5.42
XDWRITE (32)	7Fh/0004h	O	yes	5.43
XDWRITEREAD (10)	53h	O	yes	5.44
XDWRITEREAD (32)	7Fh/0007h	O	yes	5.45
XPWRITE (10)	51h	O	yes	5.46
XPWRITE (32)	7Fh/0006h	O	yes	5.47
<p>The following operation codes are vendor-specific:  02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh,  10h, 11h, 13h, 14h, 19h,  20h, 21h, 22h, 23h, 24h, 26h, 27h, 29h, 2Ch, 2Dh, and  C0h to FFh.</p> <p>All operation codes for direct-access block devices not specified in this table are reserved for future standardization.</p>				
<p><sup>a</sup> Some commands are defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.  <sup>b</sup> M = command implementation is mandatory. O = command implementation is optional. X = Command implementation requirements detailed in the reference.  <sup>c</sup> Application clients should migrate from READ (6) to READ (10) (see 5.7) and from WRITE (6) to WRITE (10) (see 5.26).  <sup>d</sup> READ CAPACITY (16) is mandatory if protection information is supported and optional otherwise.  <sup>e</sup> If the sccs bit is set to one in the standard INQUIRY data (see SPC-4), then these commands shall be supported as required by SCC-2. If the sccs bit is set to zero, then these commands shall not be supported.  <sup>f</sup> This command shall be supported if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-4) and may be supported otherwise.</p>				

The commands for direct-access block devices that are obsolete are listed in table 14.

**Table 14 — Obsolete commands for direct-access block devices** (part 1 of 2)

Command name	Operation code
CHANGE DEFINITION	40h
COMPARE	39h
COPY	18h
COPY AND VERIFY	3Ah
LOCK UNLOCK CACHE (10)	36h
LOCK UNLOCK CACHE (16)	92h
MOVE MEDIUM ATTACHED	A7h
READ ELEMENT STATUS ATTACHED	B4h
REBUILD (16)	81h

**Table 14 — Obsolete commands for direct-access block devices** (part 2 of 2)

Command name	Operation code
REGENERATE (16)	82h
RELEASE (6)	17h
RESERVE (10)	56h
RESERVE (6)	16h
RELEASE (10h)	57h
REZERO	01h
SEARCH DATA EQUAL (10)	31h
SEARCH DATA HIGH (10)	30h
SEARCH DATA LOW (10)	32h
SEEK (6)	0Bh
SEEK (10)	2Bh
SET LIMITS (10)	33h
SET LIMITS (12)	B3h
XDWRITE EXTENDED (16)	80h

## 5.2 FORMAT UNIT command

### 5.2.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 15) requests that the device server format the medium into application client accessible logical blocks as specified in the number of logical blocks and logical block length values received in the last mode parameter block descriptor (see 6.3.2) in a MODE SELECT command (see SPC-4). In addition, the device server may certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

If a device server receives a FORMAT UNIT command before receiving a MODE SELECT command with a mode parameter block descriptor, then the device server shall use the number of logical blocks and logical block length at which the logical unit is currently formatted (i.e., no change is made to the number of logical blocks and the logical block length of the logical unit during the format operation).

If any deferred downloaded code has been received as a result of a WRITE BUFFER command (see SPC-4), then that deferred downloaded code shall replace the current operational code.

**Table 15 — FORMAT UNIT command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (04h)							
1		FMTPINFO		LONGLIST	FMTDATA	CMPLST	DEFECT LIST FORMAT		
2		Vendor-specific							
3		Obsolete							
4									
5		CONTROL							

The simplest form of the FORMAT UNIT command (i.e., a FORMAT UNIT command with no parameter data) accomplishes medium formatting with little application client control over defect management. The device server implementation determines the degree of defect management that is to be performed. Additional forms

of this command increase the application client's control over defect management. The application client may specify:

- a) defect list(s) to be used;
- b) defect locations;
- c) that logical unit certification be enabled; and
- d) exception handling in the event that defect lists are not accessible.

While performing a format operation, the device server shall terminate all commands except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS commands, or a REQUEST SENSE command, then the device server shall process the command. The device server shall return data for an INQUIRY command based on the condition of the SCSI target device before beginning the FORMAT UNIT command (i.e., INQUIRY data shall not change until after successful completion of a format operation). The processing of commands in the task set when a FORMAT UNIT command is received is vendor-specific.

The PROGRESS INDICATION field in parameter data returned by the device server in response to a REQUEST SENSE command (see SPC-4) may be used by the application client at any time during a format operation to poll the logical unit's progress. While a format operation is in progress unless an error has occurred, a device server shall respond to a REQUEST SENSE command by returning parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS with the sense key specific bytes set for progress indication (see SPC-4).

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 15.

The format protection information (FMTPINFO) field (see table 20) in combination with the PROTECTION FIELD USAGE field (see 5.2.2.2) specifies whether or not the device server enables or disables the use of protection information.

Following a successful format, the P\_TYPE field in the READ CAPACITY (16) parameter data (see 5.13.1) indicates the type of protection currently in effect on the logical unit.

When protection information is written during a FORMAT UNIT command (i.e., the FMTPINFO field is set to a value greater than zero), protection information shall be written to a default value of FFFF\_FFFF\_FFFF\_FFFFh.

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as defined in table 18. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as defined in table 19. If the FMTDATA bit is set to zero, then the LONGLIST bit shall be ignored.

A format data (FMTDATA) bit set to zero specifies that no parameter list be transferred from the data-out buffer.

A FMTDATA bit set to one specifies that the FORMAT UNIT parameter list (see table 17) shall be transferred from the data-out buffer. The parameter list consists of a parameter list header, followed by an optional initialization pattern descriptor, followed by an optional defect list.

A complete list (CMPLST) bit set to zero specifies that the defect list included in the FORMAT UNIT parameter list shall be used in an addition to the existing list of defects. As a result, the device server shall construct a new GLIST (see 4.9) that contains:

- a) the existing GLIST;
- b) the DLIST, if it is sent by the application client; and
- c) the CLIST, if certification is enabled (i.e., the device server may add any defects it detects during the format operation).

A CMPLST bit set to one specifies that the defect list included in the FORMAT UNIT parameter list is a complete list of defects. Any existing defect list except the PLIST shall be ignored by the device server. As a result, the device server shall construct a new GLIST (see 4.9) that contains:

- a) the DLIST, if it is sent by the application client; and
- b) the CLIST, if certification is enabled (i.e., the device server may add any defects it detects during the format operation).

If the FMTDATA bit is set to zero, then the Cmplst bit shall be ignored.

If the FMTDATA bit is set to one, then the DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list (see table 16).

Table 16 defines the address descriptor usage for the FORMAT UNIT command.

The contents of the CONTROL byte are defined in SAM-4.

**Table 16 — FORMAT UNIT command address descriptor usage**

Field in the FORMAT UNIT CDB			DEFECT LIST LENGTH field in the parameter list header	Type <sup>a</sup>	Comments <sup>f</sup>
FMTDATA	CMPLST	DEFECT LIST FORMAT			
0	any	000b	Not available	M	Vendor-specific defect information
1	0	000b (short block)	Zero	O	See <sup>b</sup> and <sup>d</sup>
	1			O	See <sup>b</sup> and <sup>e</sup>
	0		Nonzero	O	See <sup>c</sup> and <sup>d</sup>
	1			O	See <sup>b</sup> and <sup>e</sup>
1	0	011b (long block)	Zero	O	See <sup>b</sup> and <sup>d</sup>
	1			O	See <sup>b</sup> and <sup>e</sup>
	0		Nonzero	O	See <sup>c</sup> and <sup>d</sup>
	1			O	See <sup>c</sup> and <sup>e</sup>
1	0	100b (bytes from index)	Zero	O	See <sup>b</sup> and <sup>d</sup>
	1			O	See <sup>b</sup> and <sup>e</sup>
	0		Nonzero	O	See <sup>c</sup> and <sup>d</sup>
	1			O	See <sup>c</sup> and <sup>e</sup>
1	0	101b (physical sector)	Zero	O	See <sup>b</sup> and <sup>d</sup>
	1			O	See <sup>b</sup> and <sup>e</sup>
	0		Nonzero	O	See <sup>c</sup> and <sup>d</sup>
	1			O	See <sup>c</sup> and <sup>e</sup>
1	0	110b (vendor- specific)	Vendor-specific	O	
	1			O	
All others				Reserved.	
<sup>a</sup> M = implementation is mandatory. O = implementation is optional. <sup>b</sup> No DLIST is included in the parameter list. <sup>c</sup> A DLIST is included in the parameter list. The device server shall add the DLIST defects to the new GLIST. <sup>d</sup> The device server shall add existing GLIST defects to the new GLIST (i.e., use the existing GLIST). <sup>e</sup> The device server shall not add existing GLIST defects to the new GLIST (i.e., discard the existing GLIST). <sup>f</sup> All the options described in this table cause a new GLIST to be created during processing of the FORMAT UNIT command as described in the text.					

## 5.2.2 FORMAT UNIT parameter list

### 5.2.2.1 FORMAT UNIT parameter list overview

Table 17 defines the FORMAT UNIT parameter list.

**Table 17 — FORMAT UNIT parameter list**

Bit Byte	7	6	5	4	3	2	1	0
0 to 3 or 0 to 7	Parameter list header (see table 18 or table 19 in 5.2.2.2)							
	Initialization pattern descriptor (if any)(see table 21 in 5.2.2.3)							
	Defect list (if any)							

The parameter list header is defined in 5.2.2.2.

The initialization pattern descriptor, if any, is defined in 5.2.2.3.

The defect list, if any, contains address descriptors (see 5.2.2.4) each specifying a location on the medium that the device server shall exclude from the application client accessible part. This is called the DLIST (see 4.9). The device server shall maintain the current logical block to physical block alignment (see 4.5) for logical blocks not specified in the defect list.

### 5.2.2.2 Parameter list header

The parameter list headers (see table 18 and table 19) provide several optional format control parameters. Device servers that implement these headers provide the application client additional control over the use of the four defect sources, and the format operation. If the application client attempts to select any function not implemented by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the LONGLIST bit is set to zero in the FORMAT UNIT CDB, then the short parameter list header (see table 18) is used.

**Table 18 — Short parameter list header**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved					PROTECTION FIELD USAGE		
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor-specific
2	(MSB)							
3	DEFECT LIST LENGTH							(LSB)



If the **LONGLIST** bit is set to one in the **FORMAT UNIT CDB**, then the long parameter list header (see table 19) is used.

**Table 19 — Long parameter list header**

Byte	Bit	7	6	5	4	3	2	1	0
0		Reserved					PROTECTION FIELD USAGE		
1		FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor-specific
2		Reserved							
3		Reserved							
4		(MSB)							
7		DEFECT LIST LENGTH							
		(LSB)							

The **PROTECTION FIELD USAGE** field in combination with the **FMT PINFO** field (see table 20) specifies the requested protection type (see 4.17.2).

**Table 20 — FMT PINFO field and PROTECTION FIELD USAGE field (part 1 of 2)**

Device server indication		Application client specification		Description
SPT <sup>a</sup>	PROTECT <sup>b</sup>	FMT PINFO	PROTECTION FIELD USAGE	
xxx <sup>b</sup>	0	00 <sup>b</sup>	000 <sup>b</sup>	The logical unit shall be formatted to type 0 protection <sup>c</sup> (see 4.17.2.2) resulting in the <b>P_TYPE</b> field <sup>d</sup> being set to 000 <sup>b</sup> .
xxx <sup>b</sup>	0	00 <sup>b</sup>	>000 <sup>b</sup>	Illegal <sup>e</sup>
xxx <sup>b</sup>	0	01 <sup>b</sup>	xxx <sup>b</sup>	Illegal <sup>f</sup>
xxx <sup>b</sup>	0	1x <sup>b</sup>	xxx <sup>b</sup>	Illegal <sup>f</sup>
xxx <sup>b</sup>	1	00 <sup>b</sup>	000 <sup>b</sup>	The logical unit shall be formatted to type 0 protection <sup>c</sup> (see 4.17.2.2) resulting in the <b>P_TYPE</b> field <sup>d</sup> being set to 000 <sup>b</sup> .
xxx <sup>b</sup>	1	00 <sup>b</sup>	>000 <sup>b</sup>	Illegal <sup>e</sup>
xxx <sup>b</sup>	1	01 <sup>b</sup>	xxx <sup>b</sup>	Illegal <sup>f</sup>

<sup>a</sup> See the Extended **INQUIRY** Data VPD page (see SPC-4) for the definition of the **SPT** field.

<sup>b</sup> See the standard **INQUIRY** data (see SPC-4) for the definition of the **PROTECT** bit.

<sup>c</sup> The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4).

<sup>d</sup> See the **READ CAPACITY** command (see 5.13.1) for the definition of the **P\_TYPE** field.

<sup>e</sup> The device server shall terminate the command with **CHECK CONDITION** status with the sense key set to **ILLEGAL REQUEST** and the additional sense code set to **INVALID FIELD IN PARAMETER LIST**.

<sup>f</sup> The device server shall terminate the command with **CHECK CONDITION** status with the sense key set to **ILLEGAL REQUEST** and the additional sense code set to **INVALID FIELD IN CDB**.

<sup>g</sup> The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the logical block length is 512, then the formatted logical block length is 520). Following a successful format, the **PROT\_EN** bit in the **READ CAPACITY** (16) parameter data (see 5.13.1) indicates whether protection information (see 4.17) is enabled.

Table 20 — FMTPINFO field and PROTECTION FIELD USAGE field (part 2 of 2)

Device server indication		Application client specification		Description
SPT <sup>a</sup>	PROTECT <sup>b</sup>	FMTPINFO	PROTECTION FIELD USAGE	
000b 001b 011b	1	10b	000b	The logical unit shall be formatted to type 1 protection <sup>g</sup> (see 4.17.2.3) resulting in the P_TYPE field <sup>d</sup> being set to 000b.
000b 001b 011b	1	10b	>000b	Illegal <sup>e</sup>
000b	1	11b	xxxb	Illegal <sup>f</sup>
001b	1	11b	000b	The logical unit shall be formatted to type 2 protection <sup>g</sup> (see 4.17.2.4) resulting in the P_TYPE field <sup>d</sup> being set to 001b.
001b	1	11b	>000b	Illegal <sup>e</sup>
011b	1	11b	000b	Illegal <sup>e</sup>
011b	1	1	001b	The logical unit shall be formatted to type 3 protection. <sup>g</sup> (see 4.17.2.5) resulting in the P_TYPE field <sup>d</sup> being set to 010b.
011b	1	11b	>001b	Illegal <sup>e</sup>
010b	1	1xb	xxxb	Reserved
1xxb	1	1xb	xxxb	Reserved
<sup>a</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definition of the SPT field. <sup>b</sup> See the standard INQUIRY data (see SPC-4) for the definition of the PROTECT bit. <sup>c</sup> The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4). <sup>d</sup> See the READ CAPACITY command (see 5.13.1) for the definition of the P_TYPE field. <sup>e</sup> The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. <sup>f</sup> The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>g</sup> The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the logical block length is 512, then the formatted logical block length is 520). Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 5.13.1) indicates whether protection information (see 4.17) is enabled.				

A format options valid (FOV) bit set to zero specifies that the device server shall use its default settings for the DPROY bit, the DCRT bit, the STPF bit, and IP bit. If the FOV bit is set to zero, then the application client shall set these bits to zero. If the FOV bit is set to zero, and any of the other bits listed in this paragraph are not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A FOV bit set to one specifies that the device server shall examine the values of the DPROY, DCRT, STPF, and IP bits. When the FOV bit is set to one, the DPROY, DCRT, STPF, and IP bits are defined as follows.

A disable primary (DPROY) bit set to zero specifies that the device server shall not use parts of the medium identified as defective in the PLIST for application client accessible logical blocks. If the device server is not able to locate the PLIST or it is not able to determine whether a PLIST exists, then the device server shall take the action specified by the STPF bit.

A DPROY bit set to one specifies that the device server shall not use the PLIST to identify defective areas of the medium. The PLIST shall not be deleted.

A disable certification (DCRT) bit set to zero specifies that the device server shall perform a vendor-specific medium certification operation to generate a CLIST. A DCRT bit set to one specifies that the device server shall not perform any vendor-specific medium certification process or format verification operation.

The stop format (STPF) bit controls the behavior of the device server if one of the following events occurs:

- a) The device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMLST bit is set to zero) and the device server is not able to locate the list or determine whether the list exists; or
- b) The device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMLST bit is set to zero), and the device server encounters an error while accessing the defect list.

A STPF bit set to zero specifies that, if one or both of these events occurs, the device server shall continue to process the FORMAT UNIT command. The device server shall terminate the FORMAT UNIT command with CHECK CONDITION status at the completion of the command with the sense key set to RECOVERED ERROR and the additional sense code set to either DEFECT LIST NOT FOUND if the condition described in item a) occurred, or DEFECT LIST ERROR if the condition described in item b) occurred.

A STPF bit set to one specifies that, if one or both of these events occurs, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status and the sense key shall be set to MEDIUM ERROR with the additional sense code set to either DEFECT LIST NOT FOUND if the condition described in item a) occurred, or DEFECT LIST ERROR if the condition described in item b) occurred.

NOTE 8 - The use of the FMTDATA bit, the CMLST bit, and the parameter list header allow the application client to control the source of the defect lists used by the FORMAT UNIT command. Setting the DEFECT LIST LENGTH field to zero allows the application client to control the use of PLIST and CLIST without having to specify a DLIST.

An initialization pattern (IP) bit set to zero specifies that an initialization pattern descriptor is not included and that the device server shall use its default initialization pattern. An IP bit set to one specifies that an initialization pattern descriptor (see 5.2.2.3) is included in the FORMAT UNIT parameter list following the parameter list header.

An immediate (IMMED) bit set to zero specifies that the device server shall return status after the format operation has completed. An IMMED bit value set to one specifies that the device server shall return status after the entire parameter list has been transferred.

The DEFECT LIST LENGTH field specifies the total length in bytes of the defect list (i.e., the address descriptors) that follows and does not include the initialization pattern descriptor, if any. The formats for the address descriptor(s) are shown in 5.2.2.4.

Short block format address descriptors and long block format address descriptors should be in ascending order. Bytes from index format address descriptors and physical sector format address descriptors shall be in ascending order. More than one physical or logical block may be affected by each address descriptor. If the address descriptors are not in the required order, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

### 5.2.2.3 Initialization pattern descriptor

The initialization pattern descriptor specifies that the device server initialize logical blocks to a specified pattern. The initialization pattern descriptor (see table 21) is sent to the device server as part of the FORMAT UNIT parameter list.

**Table 21 — Initialization pattern descriptor**

Bit								
Byte	7	6	5	4	3	2	1	0
0	IP MODIFIER		SI	Reserved				
1	INITIALIZATION PATTERN TYPE							
2	(MSB)INITIALIZATION PATTERN LENGTH (n - 3)							
3	(LSB)							
4	INITIALIZATION PATTERN							
n								

The initialization pattern modifier (IP MODIFIER) field (see table 22) specifies the type and location of a header that modifies the initialization pattern.

**Table 22 — Initialization pattern modifier (IP MODIFIER) field**

Code	Description
00b	No header. The device server shall not modify the initialization pattern.
01b	The device server shall overwrite the initialization pattern to write the LBA in the first four bytes of each logical block. The LBA shall be written with the most significant byte first. If the LBA is larger than four bytes, then the least significant four bytes shall be written ending with the least significant byte.
10b	The device server shall overwrite the initialization pattern to write the LBA in the first four bytes of each physical block contained within the logical block. The lowest numbered logical block or part thereof that occurs within the physical block is used. The LBA shall be written with the most significant byte first. If the LBA is larger than four bytes, then the least significant four bytes shall be written ending with the least significant byte.
11b	Reserved.

A security initialize (SI) bit set to one specifies that the device server shall attempt to write the initialization pattern to all areas of the medium including those that may have been reassigned (i.e., are in a defect list). An SI bit set to one shall take precedence over any other FORMAT UNIT CDB field. The initialization pattern shall be written using a security erasure write technique. Application clients may choose to use this command multiple times to fully erase the previous data. Such security erasure write technique procedures are outside the scope of this standard. The exact requirements placed on the security erasure write technique are vendor-specific. The intent of the security erasure write is to render any previous user data unrecoverable by any analog or digital technique.

An SI bit set to zero specifies that the device server shall initialize the application client accessible part of the medium. The device server is not required to initialize other areas of the medium. However, the device server shall format the medium as defined in the FORMAT UNIT command.

When the SI bit is set to one, the device server need not write the initialization pattern over the header and other parts of the medium not previously accessible to the application client. If the device server is unable to write over any part of the medium that is currently accessible to the application client or may be made accessible to the application client in the future (e.g., by clearing the defect list), then the device server shall

terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to the appropriate value for the condition. The device server shall attempt to rewrite all remaining parts of the medium even if some parts are not able to be rewritten.

The INITIALIZATION PATTERN TYPE field (see table 23) specifies the type of pattern the device server shall use to initialize each logical block within the application client accessible part of the medium. All bytes within a logical block shall be written with the initialization pattern. The initialization pattern is modified by the IP MODIFIER field as described in table 22.

**Table 23 — INITIALIZATION PATTERN TYPE field**

Code	Description
00h	Use a default initialization pattern <sup>a</sup>
01h	Repeat the pattern specified in the INITIALIZATION PATTERN field as required to fill the logical block <sup>b</sup>
02h to 7Fh	Reserved
80h to FFh	Vendor-specific
<sup>a</sup> If the INITIALIZATION PATTERN LENGTH field is not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. <sup>b</sup> If the INITIALIZATION PATTERN LENGTH field is set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.	

The INITIALIZATION PATTERN LENGTH field specifies the number of bytes contained in the INITIALIZATION PATTERN field. If the initialization pattern length exceeds the current logical block length, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The INITIALIZATION PATTERN field specifies the initialization pattern. The initialization pattern is modified by the IP MODIFIER field.

#### 5.2.2.4 Address descriptor formats

##### 5.2.2.4.1 Address descriptor formats overview

This subclause describes the address descriptor formats used in the FORMAT UNIT command, the READ DEFECT DATA commands (see 5.14 and 5.15), and the Translate Address diagnostic pages (see 6.1.2 and 6.1.3) of the SEND DIAGNOSTIC command and the RECEIVE DIAGNOSTIC RESULTS command.

The format type of an address descriptor is specified with:

- a) the DEFECT LIST FORMAT field in the CDB, for the FORMAT UNIT command and the READ DEFECT DATA commands;
- b) the SUPPLIED FORMAT field, for the Translate Address diagnostic pages; or
- c) the TRANSLATE FORMAT field, for the Translate Address diagnostic pages.

Table 24 defines the types of address descriptors.

**Table 24 — Address descriptor formats**

Format type	Description	Reference
000b	Short block format address descriptor	5.2.2.4.2
011b	Long block format address descriptor	5.2.2.4.3
100b	Bytes from index format address descriptor	5.2.2.4.4
101b	Physical sector format address descriptor	5.2.2.4.5
110b	Vendor-specific	
All others	Reserved	

#### 5.2.2.4.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor defined in table 25.

**Table 25 — Short block format address descriptor (000b)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)							
3	SHORT BLOCK ADDRESS (LSB)							

For the FORMAT UNIT command, the SHORT BLOCK ADDRESS field contains the four-byte LBA of a defect. If multiple logical blocks are contained within a physical block, then the device server may consider logical blocks in addition to the one specified by this descriptor as containing defects.

For the READ DEFECT DATA commands, the SHORT BLOCK ADDRESS field contains a vendor-specific four-byte value.

For the Translate Address diagnostic pages, the SHORT BLOCK ADDRESS field contains a four-byte LBA or a vendor-specific four byte value that is greater than the capacity of the medium.

#### 5.2.2.4.3 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor defined in table 26.

**Table 26 — Long block format address descriptor (011b)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)							
7	LONG BLOCK ADDRESS (LSB)							

For the FORMAT UNIT command, the LONG BLOCK ADDRESS field contains the eight-byte LBA of a defect. If multiple logical blocks are contained within a physical block, then the device server may consider logical blocks in addition to the one specified by this descriptor as containing defects.

For the READ DEFECT DATA commands, the LONG BLOCK ADDRESS field contains a vendor-specific eight-byte value.

For the Translate Address diagnostic pages, the LONG BLOCK ADDRESS field contains a eight-byte LBA or a vendor-specific eight-byte value that is greater than the capacity of the medium.

#### 5.2.2.4.4 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index address descriptor defined in table 27. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect that is either the length of one track or is no more than eight bytes long. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or the first byte or last byte of an area.

**Table 27 — Bytes from index format address descriptor (100b)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB) _____							
2	CYLINDER NUMBER _____ (LSB)							
3	HEAD NUMBER _____							
4	(MSB) _____							
7	BYTES FROM INDEX _____ (LSB)							

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

The BYTES FROM INDEX field contains the number of bytes from the index (e.g., from the start of the track) to the location being described. A BYTES FROM INDEX field set to FFFF\_FFFFh specifies that the entire track is being described.

For sorting bytes from index format address descriptors, the cylinder number is the most significant part of the address and the bytes from index is the least significant part of the address. More than one logical block may be described by this descriptor.

#### 5.2.2.4.5 Physical sector format address descriptor

A format type of 101b specifies the physical sector address descriptor defined in table 28. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect that is either the length of one track or the length of one sector. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or a sector.

**Table 28 — Physical sector format address descriptor (101b)**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB) _____							
2	CYLINDER NUMBER _____ (LSB)							
3	HEAD NUMBER _____							
4	(MSB) _____							
7	SECTOR NUMBER _____ (LSB)							

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

The SECTOR NUMBER field contains the sector number. A SECTOR NUMBER field set to FFFF\_FFFFh specifies that the entire track is being described.

For sorting physical sector format address descriptors, the cylinder number is the most significant part of the address and the sector number is the least significant part of the address. More than one logical block may be described by this descriptor.

### 5.3 ORWRITE command

The ORWRITE command (see table 29) requests that the device server perform the following as an uninterrupted series of actions:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer;
- 3) perform an OR operation with user data contained in the logical blocks transferred from the data-out buffer and the logical blocks read, storing the data resulting from the OR operation in a buffer; and
- 4) write the logical blocks from that buffer.

Each logical block includes user data and may include protection information, based on the ORPROTECT field and the medium format.

**Table 29 — ORWRITE command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (8Bh)							
1		ORPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10	(MSB)	TRANSFER LENGTH							
13									
14		Reserved			GROUP NUMBER				
15		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 29.

See the WRITE (10) command (see 5.27) for the definitions of the FUA bit and the FUA\_NV bit. See the READ (10) command (see 5.8) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that are read, transferred from the data-out buffer, and ORed into a buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the logical block address plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The contents of the CONTROL byte are defined in SAM-4.

The device server shall:

- a) check protection information read from the medium based on the ORPROTECT field as described in table 30; and
- b) check protection information transferred from the data-out buffer based on the ORPROTECT field as described in table 31.



If the check of protection information read from the medium and check protection information transferred from the data-out buffer is successful, then the device server shall set the protection information (see 4.17) as it writes each logical block to the medium as follows:

- a) the LOGICAL BLOCK GUARD field set to a CRC properly generated (see 4.17.4) by the device server;
- b) the LOGICAL BLOCK REFERENCE TAG field set to the same LOGICAL BLOCK REFERENCE TAG field received from the data-out buffer; and
- c) the LOGICAL BLOCK APPLICATION TAG field set to the same LOGICAL BLOCK APPLICATION TAG field received from the data-out buffer.

The order of the user data and protection information checks and comparisons is vendor-specific.

The device server shall check the protection information read from the medium based on the ORPROTECT field as described in table 30.

**Table 30 — ORPROTECT field - checking protection information read from the medium (part 1 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium to check. Only user data is checked.		
001b 101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition <sup>a</sup>		
010b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition <sup>a</sup>		

**Table 30 — ORPROTECT field - checking protection information read from the medium (part 2 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
011b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition <sup>a</sup>		
100b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			

**Table 30 — ORPROTECT field - checking protection information read from the medium (part 3 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
<p><sup>a</sup> An or write operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>c</sup> The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by a method not defined by this standard.</p> <p><sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.</p> <p><sup>e</sup> If multiple errors occur, the selection of which error to report is not defined by this standard.</p> <p><sup>f</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bits.</p> <p><sup>g</sup> If the device server detects a:</p> <p>a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) is enabled; or</p> <p>b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.17.2.5) is enabled,</p> <p>then the device server shall not check any protection information in the associated logical block.</p> <p><sup>h</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 3 protection is enabled, then the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.</p>				

The device server shall check the protection information transferred from the data-out buffer based on the ORPROTECT field as described in table 31.

**Table 31 — ORPROTECT field - checking protection information from the data-out buffer (part 1 of 3)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d e</sup> , additional sense code
000b	Yes	No protection information received from application client to check		
	No	No protection information received from application client to check		
001b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		

**Table 31 — ORPROTECT field - checking protection information from the data-out buffer (part 2 of 3)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d e</sup> , additional sense code
010b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		
011b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition <sup>a</sup>		
100b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition <sup>a</sup>		
101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			

**Table 31 — ORPROTECT field - checking protection information from the data-out buffer** (part 3 of 3)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d e</sup> , additional sense code
<sup>a</sup> An or write operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge is obtained by a method not defined by this standard. <sup>d</sup> If the device server terminates the command with CHECK CONDITION status, the sense key shall be set to ABORTED COMMAND. <sup>e</sup> If multiple errors occur, the selection of which error to report is not defined by this standard. <sup>f</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG, then the device server may check the logical block reference tag. The method for acquiring this knowledge is not defined by this standard.				

## 5.4 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 32) requests that the device server transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 32.

**Table 32 — PRE-FETCH (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (34h)							
1		Reserved						IMMED	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	PREFETCH LENGTH							
8									
9		CONTROL							

An immediate (IMMED) bit set to zero specifies that status shall be returned after the operation is complete. An IMMED bit set to one specifies that status shall be returned as soon as the CDB has been validated.

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.4) accessed by this command. If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the

command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The GROUP NUMBER field specifies the group into which attributes associated with the command should be collected (see 4.18). A GROUP NUMBER field set to zero specifies that any attributes associated with the command shall not be collected into any group.

The PREFETCH LENGTH field specifies the number of contiguous logical blocks that shall be pre-fetched (i.e., transferred to the cache from the medium), starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A PREFETCH LENGTH field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be pre-fetched. Any other value specifies the number of logical blocks that shall be pre-fetched. If the LBA plus the prefetch length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server is not required to transfer logical blocks that already are contained in the cache.

The contents of the CONTROL byte are defined in SAM-4.

If the IMMED bit is set to zero and the specified logical blocks were successfully transferred to the cache, then the device server shall complete the command with CONDITION MET status.

If the IMMED bit is set to zero and the cache does not have sufficient capacity to accept all of the specified logical blocks, then the device server shall transfer to the cache as many of the specified logical blocks that fit. If these logical blocks are transferred successfully, then the device server shall complete the command with GOOD status.

If the IMMED bit is set to one and the cache has sufficient capacity to accept all of the specified logical blocks, then the device server shall complete the command with CONDITION MET status.

If the IMMED bit is set to one and the cache does not have sufficient capacity to accept all of the specified logical blocks, then the device server shall complete the command with GOOD status.

If the IMMED bit is set to zero and one or more of the specified logical blocks were not successfully transferred to the cache for reasons other than lack of cache capacity, then the device server shall terminate the command with CHECK CONDITION status with the sense key and the additional sense code set to the appropriate values. If the IMMED bit is set to one and one or more of the specified logical blocks were not successfully transferred to the cache for reasons other than lack of cache capacity, the device server shall report a deferred error (see SPC-4).

## 5.5 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 33) requests that the device server transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

**Table 33 — PRE-FETCH (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (90h)							
1		Reserved						IMMED	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10	(MSB)	PREFETCH LENGTH							
13									
14		Reserved			GROUP NUMBER				
15		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 33.

See the PRE-FETCH (10) command (see 5.4) for the definitions of the other fields in this command.

## 5.6 PREVENT ALLOW MEDIUM REMOVAL command

The PREVENT ALLOW MEDIUM REMOVAL command (see table 34) requests that the logical unit enable or disable the removal of the medium. If medium removal is prevented on any I\_T nexus that has access to the logical unit, then the logical unit shall not allow medium removal.

**Table 34 — PREVENT ALLOW MEDIUM REMOVAL command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (1Eh)							
1		Reserved							
2		Reserved							
3		Reserved							
4		Reserved						PREVENT	
5		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 34.

Table 35 defines the PREVENT field values and their meanings.

**Table 35 — PREVENT field**

Value	Description
00b	Medium removal shall be allowed.
01b	Medium removal shall be prohibited.
10b to 11b	Obsolete

The contents of the CONTROL byte are defined in SAM-4.

The prevention of medium removal shall begin when any application client issues a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 01b (i.e., medium removal prevented). The prevention of medium removal for the logical unit shall terminate after:

- a) one of the following occurs for each I\_T nexus through which medium removal had been prevented:
  - A) receipt of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 00b; or
  - B) an I\_T nexus loss;
- b) a power on;
- c) a hard reset; or
- d) a logical unit reset.

If possible, the device server shall perform a synchronize cache operation before terminating the prevention of medium removal.

If a persistent reservation or registration is being preempted by a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see SPC-4), then the equivalent of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to zero shall be processed for each the I\_T nexuses associated with the persistent reservation or registrations being preempted. This allows an application client to override the prevention of medium removal function for an initiator port that is no longer operating correctly.

While a prevention of medium removal condition is in effect, the logical unit shall inhibit mechanisms that allow removal of the medium by an operator.

## 5.7 READ (6) command

The READ (6) command (see table 36) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data but does not include protection information. The most recent data value written, or to be written, if cached, in the addressed logical blocks shall be returned.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 36.

**Table 36 — READ (6) command**

Byte	Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)								
1	Reserved				(MSB)				
2	LOGICAL BLOCK ADDRESS								
3									
4	TRANSFER LENGTH								
5	CONTROL								



The cache control bits (see 5.8) are not provided for this command. Direct-access block devices with cache may have values for the cache control bits that affect the READ (6) command; however, no default values are defined by this standard. If explicit control is required, then the READ (10) command should be used.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be read and transferred to the data-in buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be read. Any other value specifies the number of logical blocks that shall be read. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 9 - For the READ (10) command, READ (12) command, READ (16) command, and READ (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are read.

NOTE 10 - Although the READ (6) command is limited to addressing logical blocks up to a capacity of 1 GiB, for logical block lengths of 512 bytes, this command has been maintained as mandatory since some system initialization routines require that the READ (6) command be used. System initialization routines should migrate from the READ (6) command to the READ (10) command, which is capable of addressing 2 TiB with logical block lengths of 512 bytes, or the READ (16) command to address more than 2 TiB.

The contents of the CONTROL byte are defined in SAM-4.

The device server shall check the protection information read from the medium before returning status for the command as described in table 37.

**Table 37 — Protection information checking for READ (6)**

Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information <sup>e</sup>	Extended INQUIRY Data VPD page bit value <sup>d</sup>	If check fails <sup>b c</sup> , additional sense code
Yes	No	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>a</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
No		No protection information available to check		

<sup>a</sup> The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.

<sup>b</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

<sup>c</sup> If multiple errors occur, the selection of which error to report is not defined by this standard.

<sup>d</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD\_CHK bit, APP\_CHK bit, and REF\_CHK bit.

<sup>e</sup> If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF\_FFFFh, and type 3 protection (see 4.17.2.5) is enabled,

then the device server shall not check any protection information in the associated logical block.

<sup>f</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, then the device server checks the logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

## 5.8 READ (10) command

The READ (10) command (see table 38) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format. The most recent data value written in the addressed logical block shall be returned.

**Table 38 — READ (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (28h)							
1		RDPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 38.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

The device server shall check the protection information read from the medium before returning status for the command based on the RDPROTECT field as described in table 39.

**Table 39 — RDPROTECT field (part 1 of 3)**

Code	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information <sup>h</sup>	Extended INQUIRY Data VPD page bit value <sup>g</sup>	If check fails <sup>d f</sup> , additional sense code
000b	Yes	No	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>i</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No		No protection information available to check		

Table 39 — RDPROTECT field (part 2 of 3)

Code	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information <sup>h</sup>	Extended INQUIRY Data VPD page bit value <sup>g</sup>	If check fails <sup>d f</sup> , additional sense code
001b 101b <sup>b</sup>	Yes	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>i</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No <sup>a</sup>	No protection information available to transmit to the data-in buffer or for checking			
010b <sup>b</sup>	Yes	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	No check performed	
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>i</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No <sup>a</sup>	No protection information available to transmit to the data-in buffer or for checking			
	011b <sup>b</sup>	Yes	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	No check performed
LOGICAL BLOCK APPLICATION TAG				No check performed	
LOGICAL BLOCK REFERENCE TAG				No check performed	
No <sup>a</sup>		No protection information available to transmit to the data-in buffer or for checking			

Table 39 — RDPROTECT field (part 3 of 3)

Code	Logical unit formatted with protection information	Shall device server transmit protection information?	Field in protection information <sup>h</sup>	Extended INQUIRY Data VPD page bit value <sup>g</sup>	If check fails <sup>d f</sup> , additional sense code
100b <sup>b</sup>	Yes	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	No check performed	
			LOGICAL BLOCK REFERENCE TAG	No check performed	
	No <sup>a</sup>	No protection information available to transmit to the data-in buffer or for checking			
110b to 111b	Reserved				

<sup>a</sup> A read operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

<sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

<sup>c</sup> The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the READ (32) command (see 5.11) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be acquired by a method not defined by this standard.

<sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

<sup>e</sup> Transmit protection information to the data-in buffer.

<sup>f</sup> If multiple errors occur, the selection of which error to report is not defined by this standard.

<sup>g</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD\_CHK bit, the APP\_CHK bit, and the REF\_CHK bit.

<sup>h</sup> If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF\_FFFFh, and type 3 protection (see 4.17.2.5) is enabled,

then the device server shall not check any protection information in the associated logical block.

<sup>i</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, then the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a READ (32) command (see 5.11). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

A disable page out (DPO) bit set to zero specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page (see 6.3.4). A DPO bit set to one specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page. All other aspects of the algorithm implementing the cache replacement strategy are not defined by this standard.

NOTE 11 - The DPO bit is used to control replacement of logical blocks in the cache when the application client has information on the future usage of the logical blocks. If the DPO bit is set to one, then the application client is specifying that the logical blocks accessed by the command are not likely to be accessed again in the near future and should not be put in the cache nor retained by the cache. If the DPO bit is set to zero, then the application client is specifying that the logical blocks accessed by this command are likely to be accessed again in the near future.

The force unit access (FUA) and force unit access non-volatile cache (FUA\_NV) bits are defined in table 40.

**Table 40 — Force unit access for read operations**

FUA	FUA_NV	Description
0	0	The device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium.
0	1	<p>If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall read the logical blocks from non-volatile cache or the medium. If a non-volatile cache is present and a volatile cache contains a more recent version of a logical block, then the device server shall write the logical block to:</p> <ul style="list-style-type: none"> <li>a) non-volatile cache; and/or</li> <li>b) the medium,</li> </ul> <p>before reading it.</p> <p>If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-4), then the device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium.</p>
1	0 or 1	The device server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, then the device server shall write the logical block to the medium before reading it.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be read and transferred to the data-in buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be read. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be read. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 12 - For the READ (6) command, a TRANSFER LENGTH field set to zero specifies that 256 logical blocks are read.

The contents of the CONTROL byte are defined in SAM-4.

## 5.9 READ (12) command

The READ (12) command (see table 41) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

**Table 41 — READ (12) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (A8h)							
1		RDPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6	(MSB)	TRANSFER LENGTH							
9									
10	Restricted for MMC-6	Reserved			GROUP NUMBER				
11		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 41.

See the READ (10) command (see 5.8) for the definitions of the other fields in this command.

## 5.10 READ (16) command

The READ (16) command (see table 42) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

**Table 42 — READ (16) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (88h)							
1	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
2	(MSB) LOGICAL BLOCK ADDRESS (LSB)							
9								
10	(MSB) TRANSFER LENGTH (LSB)							
13								
14	Restricted for MMC-6	Reserved		GROUP NUMBER				
15	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 42.

See the READ (10) command (see 5.8) for the definitions of the other fields in this command.

## 5.11 READ (32) command

The READ (32) command (see table 43) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

The READ (32) command shall only be processed if type 2 protection is enabled (see 4.17.2.4).

**Table 43 — READ (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0009h)							
9									
10		RDPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
23									
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG							
25									
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 43.

See the READ (10) command (see 5.8) for the definitions of the CONTROL byte, GROUP NUMBER field, the RDPROTECT field, the DPO bit, the FUA bit, the FUA\_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 39 in 5.8), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 39 in 5.8), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in the



protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in the protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

- a) the ATO bit is set to zero; or
- b) the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 39 in 5.8).

## 5.12 READ CAPACITY (10) command

### 5.12.1 READ CAPACITY (10) overview

The READ CAPACITY (10) command (see table 44) requests that the device server transfer 8 bytes of parameter data describing the capacity and medium format of the direct-access block device to the data-in buffer. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.12). If the logical unit supports protection information (see 4.17), then the application client should use the READ CAPACITY (16) command instead of the READ CAPACITY (10) command.

**Table 44 — READ CAPACITY (10) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (25h)							
1	Reserved							Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								
6	Reserved							
7								
8	Reserved							PMI
9	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 44.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

The LOGICAL BLOCK ADDRESS field shall be set to zero if the PMI bit is set to zero. If the PMI bit is set to zero and the LOGICAL BLOCK ADDRESS field is not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

A partial medium indicator (PMI) bit set to zero specifies that the device server return information on the last logical block on the direct-access block device.

A PMI bit set to one specifies that the device server return information on the last logical block after that specified in the LOGICAL BLOCK ADDRESS field before a substantial vendor-specific delay in data transfer may be encountered.

NOTE 13 - This function is intended to assist storage management software in determining whether there is sufficient space starting with the LBA specified in the CDB to contain a frequently accessed data structure (e.g., a file directory or file index) without incurring an extra delay.

The contents of the CONTROL byte are defined in SAM-4.

### 5.12.2 READ CAPACITY (10) parameter data

The READ CAPACITY (10) parameter data is defined in table 45. Any time the READ CAPACITY (10) parameter data changes, the device server should establish a unit attention condition as described in 4.7.

**Table 45 — READ CAPACITY (10) parameter data**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)	RETURNED LOGICAL BLOCK ADDRESS						(LSB)
3								
4	(MSB)	LOGICAL BLOCK LENGTH IN BYTES						(LSB)
7								

If the number of logical blocks exceeds the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to FFFF\_FFFFh. The application client should then issue a READ CAPACITY (16) command (see 5.13) to retrieve the READ CAPACITY (16) parameter data.

If the PMI bit is set to zero, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the lower of:

- a) the LBA of the last logical block on the direct-access block device; or
- b) FFFF\_FFFFh.

If the PMI bit is set to one, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the lower of:

- a) the last LBA after that specified in the LOGICAL BLOCK ADDRESS field of the CDB before a substantial vendor-specific delay in data transfer may be encountered; or
- b) FFFF\_FFFFh.

The RETURNED LOGICAL BLOCK ADDRESS shall be greater than or equal to that specified by the LOGICAL BLOCK ADDRESS field in the CDB.

The LOGICAL BLOCK LENGTH IN BYTES field contains the number of bytes of user data in the logical block indicated by the RETURNED LOGICAL BLOCK ADDRESS field. This value does not include protection information or additional information (e.g., ECC bytes) recorded on the medium.

## 5.13 READ CAPACITY (16) command

### 5.13.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 46) requests that the device server transfer parameter data describing the capacity and medium format of the direct-access block device to the data-in buffer. This command is mandatory if the logical unit supports protection information (see 4.17) and is optional otherwise. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2). This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.12).

**Table 46 — READ CAPACITY (16) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (10h)				
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	ALLOCATION LENGTH							(LSB)
14	Reserved							PMI
15	CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 46.

See the READ CAPACITY (10) command (see 5.12) for definitions of the LOGICAL BLOCK ADDRESS field and the PMI bit.

The ALLOCATION LENGTH field specifies the maximum number of bytes that the application client has allocated for returned parameter data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The device server shall terminate transfers to the data-in buffer when the number of bytes specified by the ALLOCATION LENGTH field have been transferred or when all available data has been transferred, whichever is less. The contents of the parameter data shall not be altered to reflect the truncation, if any, that results from an insufficient allocation length.

The contents of the CONTROL byte are defined in SAM-4.

### 5.13.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 47. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.7.

**Table 47 — READ CAPACITY (16) parameter data**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)							
7	RETURNED LOGICAL BLOCK ADDRESS (LSB)							
8	(MSB)							
11	LOGICAL BLOCK LENGTH IN BYTES (LSB)							
12	Reserved				P_TYPE			PROT_EN
13	Reserved				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	Reserved		(MSB)					
15	LOWEST ALIGNED LOGICAL BLOCK ADDRESS (LSB)							
16								
31	Reserved							

The RETURNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are the same as the in the READ CAPACITY (10) parameter data (see 5.12). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFF\_FFFF\_FFFF\_FFEh.

The protection type (P\_TYPE) field and the protection enable (PROT\_EN) bit (see table 48) indicate the logical unit's current type of protection.

**Table 48 — P\_TYPE field and PROT\_EN bit**

PROT_EN	P_TYPE	Description
0	xxxb	The logical unit is formatted to type 0 protection (see 4.17.2.2).
1	000b	The logical unit is formatted to type 1 protection (see 4.17.2.3).
1	001b	The logical unit is formatted to type 2 protection (see 4.17.2.4).
1	010b	The logical unit is formatted to type 3 protection (see 4.17.2.5).
1	011b to 111b	Reserved

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in table 49.

**Table 49 — LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field**

Code	Description
0	One or more physical blocks per logical block <sup>a</sup>
n > 0	2 <sup>n</sup> logical blocks per physical block
<sup>a</sup> The number of physical blocks per logical block is not reported.	

The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field indicates the LBA of the first logical block that is located at the beginning of a physical block (see 4.5).

NOTE 14 - The highest LBA that the lowest aligned logical block address field supports is 3FFFh (i.e., 16 383).

## 5.14 READ DEFECT DATA (10) command

### 5.14.1 READ DEFECT DATA (10) command overview

The READ DEFECT DATA (10) command (see table 50) requests that the device server transfer the medium defect data to the data-in buffer.

**Table 50 — READ DEFECT DATA (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (37h)							
1		Reserved							
2		Reserved			REQ_PLIST	REQ_GLIST	DEFECT LIST FORMAT		
3		Reserved							
6		Reserved							
7	(MSB)	ALLOCATION LENGTH							
8		(LSB)							
9		CONTROL							

If the device server is unable to access the medium defect data, then the device server shall terminate the command with CHECK CONDITION status. If a medium error occurred, then the device server shall set the sense key to MEDIUM ERROR, or, if there is no medium defect data, then the device server shall set the sense key to NO SENSE. The device server shall set the additional sense code to DEFECT LIST NOT FOUND.

NOTE 15 - Some device servers may not be able to return medium defect data until after a FORMAT UNIT command (see 5.2) has been completed successfully.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 50.

A request primary defect list (REQ\_PLIST) bit set to zero specifies that the device server shall not return the PLIST. A REQ\_PLIST bit set to one specifies that the device server shall return the PLIST, if any.

A request grown defect list (REQ\_GLIST) bit set to zero specifies that the device server shall not return the GLIST. A REQ\_GLIST bit set to one specifies that the device server shall return the GLIST, if any.

A REQ\_PLIST bit set to zero and a REQ\_GLIST bit set to zero specifies that the device server shall return only the defect list header (i.e., the first four bytes of the defect list).

A REQ\_PLIST bit set to one and a REQ\_GLIST bit set to one specifies that the device server shall return both the PLIST and GLIST, if any. The order the lists are returned in is vendor-specific. Whether the lists are merged or not is vendor-specific.

The DEFECT LIST FORMAT field specifies the preferred format for the defect list. This field is intended for those device servers capable of returning more than one format, as defined in the FORMAT UNIT command (see 5.2.2.4). A device server unable to return the requested format shall return the defect list in its default format and indicate that format in the DEFECT LIST FORMAT field in the defect list header (see table 51).

If the requested defect list format and the returned defect list format are not the same, then the device server shall transfer the defect data and then terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to DEFECT LIST NOT FOUND.

The ALLOCATION LENGTH field is defined in the READ CAPACITY (16) command (see 5.13). The application client is responsible for comparing the allocation length requested in the CDB with the defect list length returned in the parameter data to determine whether a partial list was received. If the number of address descriptors the device server has to report exceeds the maximum value that is able to be specified in the ALLOCATION LENGTH field, then the device server shall transfer no data and shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The contents of the CONTROL byte are defined in SAM-4.

#### 5.14.2 READ DEFECT DATA (10) parameter data

The READ DEFECT DATA (10) parameter data (see table 51) contains a four-byte header, followed by zero or more address descriptors.

**Table 51 — READ DEFECT DATA (10) parameter data**

Bit	7	6	5	4	3	2	1	0
Byte								
0	Reserved							
1	Reserved			PLISTV	GLISTV	DEFECT LIST FORMAT		
2	(MSB)							
3	DEFECT LIST LENGTH (n - 3)							(LSB)
Defect list (if any)								
4	Address descriptor(s) (if any)							
n								

A PLIST valid (PLISTV) bit set to zero indicates that the data returned does not contain the PLIST. A PLISTV bit set to one indicates that the data returned contains the PLIST.

A GLIST valid (GLISTV) bit set to zero indicates that the data returned does not contain the GLIST. A GLISTV bit set to one indicates that the data returned contains the GLIST.

The DEFECT LIST FORMAT field indicates the format of the address descriptors returned by the device server. This field is defined in the FORMAT UNIT command (see 5.2.2.4).

If the device server returns short block format address descriptors (see 5.2.2.4.2) or long block format address descriptors (see 5.2.2.4.3), then the address descriptors contain vendor-specific values.

**NOTE 16** - The use of the short block format and the long block format is not recommended for this command. There is no standard model that defines the meaning of the block address of a defect. In the usual case, a defect that has been reassigned no longer has an LBA.

If the device server returns physical sector format address descriptors (see 5.2.2.4.5), then the device server may or may not include defects in parts of the medium not accessible to the application client. If the device server returns bytes from index format address descriptors (see 5.2.2.4.4), then the device server shall return a complete list of the defects. A complete list of the defects may include defects in areas not within the capacity returned in the READ CAPACITY command.

The DEFECT LIST LENGTH field indicates the length in bytes of the address descriptors that follow. The DEFECT LIST LENGTH is equal to four or eight times the number of the address descriptors, depending on the format of the returned address descriptors (see 5.2.2.4).

The address descriptors may or may not be sent in ascending order.

## 5.15 READ DEFECT DATA (12) command

### 5.15.1 READ DEFECT DATA (12) command overview

The READ DEFECT DATA (12) command (see table 52) requests that the device server transfer the medium defect data to the data-in buffer.

**Table 52 — READ DEFECT DATA (12) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (B7h)							
1	Reserved			REQ_PLIST	REQ_GLIST	DEFECT LIST FORMAT		
2	Reserved							
5								
6								
6	(MSB)	ALLOCATION LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 52.

See the READ DEFECT DATA (10) command (see 5.14) for the definitions of the other fields in this command.

NOTE 17 - The application client may determine the length of the defect list by sending the READ DEFECT DATA (12) command with an ALLOCATION LENGTH field set to eight. The device server returns the defect list header that contains the length of the defect list.

### 5.15.2 READ DEFECT DATA (12) parameter data

The READ DEFECT DATA (12) parameter data (see table 53) contains an eight byte header, followed by zero or more address descriptors.

**Table 53 — READ DEFECT DATA (12) parameter data**

Byte	Bit	7	6	5	4	3	2	1	0
0	Reserved								
1	Reserved			PLISTV		GLISTV		DEFECT LIST FORMAT	
2	Reserved								
3	Reserved								
4	(MSB)	DEFECT LIST LENGTH (n - 7)							
7									
Defect list (if any)									
8	Address descriptor(s) (if any)								
n									

See the READ DEFECT DATA (10) command (see 5.14) for the definitions of the fields in the defect list.

## 5.16 READ LONG (10) command

The READ LONG (10) command (see table 54) requests that the device server transfer data from a single logical block or physical block to the data-in buffer. The data transferred during the READ LONG (10) command is vendor-specific, but shall include the following items recorded on the medium:

- a) if a logical block is being transferred, then:
  - A) user data or transformed user data for the logical block;
  - B) protection information or transformed protection information, if any, for the logical block; and
  - C) any additional information (e.g., ECC bytes) for all the physical blocks in the logical block.
- or
- b) if a physical block is being transferred, then:
  - A) user data or transformed user data for all the logical blocks in the physical block;
  - B) protection information or transformed protection information, if any, for all the logical blocks in the physical block; and
  - C) any additional information (e.g., ECC bytes);.

If a cache contains a more recent version of the specified logical block or physical block, then the device server shall write the logical block or physical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.3.5) do not apply to this command. The device server may perform retries while processing this command.

**Table 54 — READ LONG (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (3Eh)							
1		Reserved					PBLOCK	CORRCT	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5		(LSB)							
6		Reserved							
7	(MSB)	BYTE TRANSFER LENGTH							
8		(LSB)							
9		CONTROL							

The LOGICAL BLOCK ADDRESS field specifies an LBA (see 4.4). If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the additional information contain an ECC, then any other additional bytes that are correctable by ECC should be included (e.g., a data synchronization mark within the area covered by ECC). It is not required for the ECC bytes to be at the end of the user data or protection information, if any. However, the ECC bytes should be in the same order as they are on the medium.

If there is more than one logical block per physical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) data (see 5.13.1) is set to a non-zero value), then:

- a) the device server shall support the physical block (PBLOCK) bit;
- b) a PBLOCK bit set to one specifies that the device server shall return the entire physical block containing the specified logical block; and
- c) a PBLOCK bit set to zero specifies that the device server shall return bytes representing only the specified logical block.

If there are one or more physical blocks per logical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) data (see 5.13.1) is set to zero), and the PBLOCK bit is set to one,



then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 54.

A correct (CORRECT) bit set to zero specifies that a logical block be read without any correction made by the device server. A CORRECT bit set to one should result in the device server completing the command with GOOD status unless data is not transferred for some reason other than that the data is non-correctable. In this case the device server shall complete or terminate the command with the appropriate status and sense data. A CORRECT bit set to one specifies that the data be corrected by ECC before being transferred to the data-in buffer.

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that shall be read from the specified logical block or physical block and transferred to the data-in buffer. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the available data length, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.14 and SPC-4), the VALID and ILI bits shall each be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested byte transfer length minus the actual available data length in bytes. Negative values shall be indicated by two's complement notation.

A BYTE TRANSFER LENGTH field set to zero specifies that no bytes shall be read. This condition shall not be considered an error.

The contents of the CONTROL byte are defined in SAM-4.

## 5.17 READ LONG (16) command

The READ LONG (16) command (see table 55) requests that the device server transfer data from a single logical block or physical block to the data-in buffer. The data transferred during the READ LONG (16) command is vendor-specific, but shall include the following items recorded on the medium:

- a) if a logical block is being transferred, then:
  - A) user data or transformed user data for the logical block;
  - B) protection information or transformed protection information, if any, for the logical block; and
  - C) any additional information (e.g., ECC bytes) for all the physical blocks in the logical block.
 or
- b) if a physical block is being transferred, then:
  - A) user data or transformed user data for all the logical blocks in the physical block;
  - B) protection information or transformed protection information, if any, for all the logical blocks in the physical block; and
  - C) any additional information (e.g., ECC bytes).

If a cache contains a more recent version of the specified logical block or physical block, then the device server shall write the logical block or physical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.3.5) do not apply to this command. The device server may

perform retries while processing this command. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2).

**Table 55 — READ LONG (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (9Eh)							
1		Reserved			SERVICE ACTION (11h)				
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10		Reserved							
11									
12	(MSB)	BYTE TRANSFER LENGTH							
13									
14		Reserved						PBLOCK	CORRCT
15		CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-4 and shall be set to the value defined in table 55.

See the READ LONG (10) command (see 5.16) for the definitions of the fields in this command.

## 5.18 REASSIGN BLOCKS command

### 5.18.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 56) requests that the device server reassign defective logical blocks to another area on the medium set aside for this purpose. The device server should also record the location of the defective logical blocks in the GLIST, if supported. This command shall not alter the contents of the PLIST (see 4.9).

The parameter list provided in the data-out buffer contains a defective LBA list that contains the LBAs of the logical blocks to be reassigned. The device server shall reassign the parts of the medium used for each logical block in the defective LBA list. More than one physical block may be relocated by each LBA. If the device server is able to recover user data and protection information, if any, from the original logical block, then the device server shall write the recovered user data and any protection information to the reassigned logical block. If the device server is unable to recover user data and protection information, if any, then the device server shall write vendor-specific data as the user data and shall write a default value of FFFF\_FFFF\_FFFF\_FFFFh as the protection information, if enabled. The data in all other logical blocks on the medium shall be preserved.

NOTE 18 - The effect of specifying a logical block to be reassigned that previously has been reassigned is to reassign the logical block again. Although not likely, over the life of the medium, a logical block may be assigned to multiple physical block addresses until no more spare locations remain on the medium.

**Table 56 — REASSIGN BLOCKS command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (07h)							
1		Reserved						ONGLBA	ONGLIST
2		Reserved							
4									
5		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 56.

A long LBA (ONGLBA) bit set to zero specifies that the REASSIGN BLOCKS defective LBA list contains four byte LBAs. A ONGLBA bit set to one specifies that the REASSIGN BLOCKS defective LBA list contains eight byte LBAs.

The contents of the CONTROL byte are defined in SAM-4.

### 5.18.2 REASSIGN BLOCKS parameter list

The REASSIGN BLOCKS parameter list (see table 57) contains a four-byte parameter list header followed by a defective LBA list containing one or more LBAs.

**Table 57 — REASSIGN BLOCKS parameter list**

Byte	Bit	7	6	5	4	3	2	1	0
0		Parameter list header (see table 58 or table 59)							
3									
4		DEFECTIVE LBA LIST (if any)							
n									

If ONGLIST is set to zero, then the parameter list header is defined in table 58.

**Table 58 — REASSIGN BLOCKS short parameter list header**

Byte	Bit	7	6	5	4	3	2	1	0
0		Reserved							
1									
2		(MSB)	DEFECT LIST LENGTH						
3									(LSB)

If **ONGLIST** is set to one, then the parameter list header is defined in table 59.

**Table 59 — REASSIGN BLOCKS long parameter list header**

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)							
3	DEFECT LIST LENGTH							(LSB)

The **DEFECT LIST LENGTH** field indicates the total length in bytes of the **DEFECTIVE LBA LIST** field. The **DEFECT LIST LENGTH** field does not include the parameter list header length and is equal to either:

- a) four times the number of LBAs, if the **ONGLBA** bit is set to zero; or
- b) eight times the number of LBAs, if the **ONGLBA** bit is set to one.

The **DEFECTIVE LBA LIST** field contains a list of defective LBAs. Each LBA is a four-byte field if the **ONGLBA** bit is set to zero or an eight-byte field if the **ONGLBA** bit is set to one. The LBAs shall be in ascending order.

If the direct-access block device has insufficient capacity to reassign all of the specified logical blocks, then the device server shall terminate the command with **CHECK CONDITION** status with the sense key set to **HARDWARE ERROR** and the additional sense code set to **NO DEFECT SPARE LOCATION AVAILABLE**.

If the direct-access block device is unable to successfully complete a **REASSIGN BLOCKS** command, then the device server shall terminate the command with **CHECK CONDITION** status with the appropriate sense data (see 4.14 and SPC-4). The first LBA not reassigned shall be returned in the **COMMAND-SPECIFIC INFORMATION** field of the sense data. If information about the first LBA not reassigned is not available, or if all the defects have been reassigned, then the **COMMAND-SPECIFIC INFORMATION** field shall be set to **FFFF\_FFFFh** if fixed format sense data is being used or **FFFF\_FFFF\_FFFF\_FFFFh** if descriptor format sense data is being used.

If the **REASSIGN BLOCKS** command failed due to an unexpected unrecovered read error that would cause the loss of data in a logical block not specified in the defective LBA list, then the LBA of the logical block with the unrecovered read error shall be returned in the **INFORMATION** field of the sense data and the **VALID** bit shall be set to one.

**NOTE 19** - If the device server terminates the **REASSIGN BLOCKS** command with **CHECK CONDITION** status, and the sense data **COMMAND-SPECIFIC INFORMATION** field contains a valid LBA, then the application client should remove all LBAs from the defective LBA list prior to the one returned in the **COMMAND-SPECIFIC INFORMATION** field. If the sense key is set to **MEDIUM ERROR**, and the **INFORMATION** field contains the valid LBA, then the application client should insert that new defective LBA into the defective LBA list and reissue the **REASSIGN BLOCKS** command with the new defective LBA list. Otherwise, the application client should perform any corrective action indicated by the sense data and then reissue the **REASSIGN BLOCKS** command with the new defective LBA list.

## 5.19 START STOP UNIT command

The START STOP UNIT command (see table 60) requests that the device server change the power condition of the logical unit (see 4.16) or load or eject the medium. This includes specifying that the device server enable or disable the direct-access block device for medium access operations by controlling power conditions and timers.

If any deferred downloaded code has been received as a result of a WRITE BUFFER command (see SPC-4), then that deferred downloaded code shall replace the current operational code.

**Table 60 — START STOP UNIT command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (1Bh)							
1		Reserved							IMMED
2		Reserved							
3		Reserved				POWER CONDITION MODIFIER			
4		POWER CONDITION				Reserved	NO_FLUSH	LOEJ	START
5		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 60.

If the immediate (IMMED) bit is set to zero, then the device server shall return status after the operation is completed. If the IMMED bit set to one, then the device server shall return status as soon as the CDB has been validated.

The POWER CONDITION MODIFIER field defined in table 61 is used to specify additional information about the power condition specified in the POWER CONDITION field.

**Table 61 — POWER CONDITION MODIFIER field**

POWER CONDITION field value	Code	Description
All values that are not reserved	0h	Reserved
2h (i.e., IDLE)	1h	Specifies that the device server shall increase the tolerance of the direct access block device to external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position).
	2h	Specifies that the device server shall increase the tolerance of the direct access block device to external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position) and should cause the device to use less power than when this field is set to 1h (e.g., cause a device that has rotating media to rotate the media at a lower RPM).
All other combinations		Reserved

The POWER CONDITION field is used to specify that the logical unit be placed into a power condition or to adjust a timer as defined in table 62. If this field is supported and is set to a value other than 0h, then the START and LOEJ bits shall be ignored.

**Table 62 — POWER CONDITION field**

Code	Name	Description
0h	START_VALID	Process the START and LOEJ bits.
1h	ACTIVE	Place the device into the active power condition.
2h	IDLE	Place the device into the idle power condition.
3h	STANDBY	Place the device into the standby power condition.
4h	Reserved	Reserved
5h	Obsolete	Obsolete
6h	Reserved	Reserved
7h	LU_CONTROL	Transfer control of power conditions to the logical unit.
8h to 9h	Reserved	Reserved
Ah	FORCE_IDLE_0	Force the idle condition timer to zero.
Bh	FORCE_STANDBY_0	Force the standby condition timer to zero.
Ch to Fh	Reserved	Reserved

If the START STOP UNIT command is processed with the POWER CONDITION field set to ACTIVE, IDLE, or STANDBY, then:

- the logical unit shall transition to the specified power condition; and
- the device server shall disable the idle condition timer if it is active (see SPC-4) and disable the standby condition timer if it is active (see SPC-4) until another START STOP UNIT command is processed that returns control of the power condition to the logical unit, or a logical unit reset occurs.

If the START STOP UNIT command is processed with the POWER CONDITION field set to LU\_CONTROL, then the device server shall enable the idle condition timer if it is active (see SPC-4) and disable the standby condition timer if it is active (see SPC-4).

If the START STOP UNIT command is processed with the POWER CONDITION field set to FORCE\_IDLE\_0 or FORCE\_STANDBY\_0, then the device server shall:

- force the specified timer to zero, cause the logical unit to transition to the specified power condition, and return control of the power condition to the device server; or
- terminate a START STOP UNIT command that selects a timer that is not supported by the device server or a timer that is not active. The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

It is not an error to specify that the logical unit transition to its current power condition.

If the NO\_FLUSH bit is set to zero, then logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.20 and 5.21) with the SYNC\_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to zero, and the NUMBER OF LOGICAL BLOCKS field set to zero) prior to entering into any power condition that prevents accessing the medium (e.g., before the rotating media spindle motor is stopped during transition to the stopped power condition). If the NO\_FLUSH bit is set to one, then cached logical blocks should not be written to the medium by the logical unit prior to entering into any power condition that prevents accessing the medium.

If the load eject (LOEJ) bit is set to zero, then the logical unit shall take no action regarding loading or ejecting the medium. If the LOEJ bit is set to one, then the logical unit shall unload the medium if the START bit is set to zero. If the LOEJ bit is set to one, then the logical unit shall load the medium if the START bit is set to one.

If the START bit is set to zero, then the logical unit shall:

- a) transition to the stopped power condition;
- b) disable the idle condition timer if it is active (see SPC-4); and
- c) disable the standby condition timer if it is active (see SPC-4).

If the START bit set to one, then the logical unit shall:

- a) transition to the active power condition;
- b) enable the idle condition timer if it is active; and
- c) enable the standby condition timer if it is active.

The contents of the CONTROL byte are defined in SAM-4.

## 5.20 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 63) requests that the device server ensure that the specified logical blocks have their most recent data values recorded in non-volatile cache and/or on the medium, based on the SYNC\_NV bit. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. Logical blocks may or may not be removed from volatile cache and non-volatile cache as a result of the synchronize cache operation.

**Table 63 — SYNCHRONIZE CACHE (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (35h)							
1		Reserved					SYNC_NV	IMMED	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	NUMBER OF LOGICAL BLOCKS							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 63.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

The SYNC\_NV bit (see table 64) specifies whether the device server is required to synchronize volatile and non-volatile caches.

**Table 64 — SYNC\_NV bit**

Code	Device server requirement to synchronize logical blocks currently in the	
	Volatile cache	Non-volatile cache
0	Device server shall synchronize to the medium.	Device server shall synchronize to the medium.
1	If a non-volatile cache is present, then the device server shall synchronize to non-volatile cache or the medium. If a non-volatile cache is not present, then the device server shall synchronize to the medium.	No requirement.

An immediate (IMMED) bit set to zero specifies that the device server shall not return status until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one, and the device server does not support the IMMED bit, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the LBA plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

A logical block within the range that is not in cache is not considered an error.

The contents of the CONTROL byte are defined in SAM-4.

## 5.21 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 65) requests that the device server ensure that the specified logical blocks have their most recent data values recorded in non-volatile cache and/or on the medium, based on the SYNC\_NV bit. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. Logical blocks may or may not be removed from volatile cache and non-volatile cache as a result of the synchronize cache operation.

**Table 65 — SYNCHRONIZE CACHE (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (91h)							
1		Reserved					SYNC_NV	IMMED	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9		(LSB)							
10	(MSB)	NUMBER OF LOGICAL BLOCKS							
13		(LSB)							
14		Reserved			GROUP NUMBER				
15		CONTROL							



The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 65.

See the SYNCHRONIZE CACHE (10) command (see 5.20) for the definitions of the other fields in this command.

## 5.22 VERIFY (10) command

The VERIFY (10) command (see table 66) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

**Table 66 — VERIFY (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (2Fh)							
1		VRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6	Restricted for MMC-6	Reserved			GROUP NUMBER				
7	(MSB)	VERIFICATION LENGTH							
8									
9		CONTROL							

Logical units that contain cache shall write referenced cached logical blocks to the medium for the logical unit (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.20 and 5.21) with the SYNC\_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to the value of the VERIFY command's LOGICAL BLOCK ADDRESS field, and the NUMBER OF BLOCKS field set to the value of the VERIFY command's VERIFICATION LENGTH field).

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 66.

See the READ (10) command (see 5.8) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

If the Verify Error Recovery mode page (see 6.3.6) is implemented, then the current settings in that page specify the verification criteria. If the Verify Error Recovery mode page is not implemented, then the verification criteria is vendor-specific.

If the byte check (BYTCHK) bit is set to zero, then the device server shall:

- perform a medium verification with no data comparison and not transfer any data from the data-out buffer; and
- check protection information read from the medium based on the VRPROTECT field as described in table 67.

If the BYTCHK bit is set to one, then the device server shall:

- perform a byte-by-byte comparison of user data read from the medium and user data transferred from the data-out buffer;
- check protection information read from the medium based on the VRPROTECT field as described in table 68;
- check protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 69; and

- d) perform a byte-by-byte comparison of protection information read from the medium and transferred from the data-out buffer based on the VRPROTECT field as described in table 70.

The order of the user data and protection information checks and comparisons is vendor-specific.

If a byte-by-byte comparison is unsuccessful for any reason, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to the appropriate value for the condition.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks that shall be verified, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the BYTCHK bit is set to one, then the VERIFICATION LENGTH field also specifies the number of logical blocks that the device server shall transfer from the data-out buffer. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be verified. This condition shall not be considered as an error. Any other value specifies the number of logical blocks that shall be verified. If the LBA plus the verification length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The contents of the CONTROL byte are defined in SAM-4.

If the BYTCHK bit is set to zero, then the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 67.

**Table 67 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (part 1 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium to check. Only user data is checked.		
001b 101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition <sup>a</sup>		

**Table 67 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (part 2 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
010b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition <sup>a</sup>		
011b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition <sup>a</sup>		
100b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			

**Table 67 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (part 3 of 3)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
<p><sup>a</sup> A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>c</sup> The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard.</p> <p><sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.</p> <p><sup>e</sup> If multiple errors occur, then the selection of which error to report is not defined by this standard.</p> <p><sup>f</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bits.</p> <p><sup>g</sup> If the device server detects a:</p> <p>a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or</p> <p>b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.17.2.5) is enabled,</p> <p>then the device server shall not check any protection information in the associated logical block.</p> <p><sup>h</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, then the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.</p>				

If the BYTCHK bit is set to one, then the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 68.

**Table 68 — VRPROTECT field with BYTCHK set to one - checking protection information read from the medium (part 1 of 2)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 <sup>c g</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 <sup>h</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium available to check		
001b 010b 011b 100b 101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			

**Table 68 — VRPROTECT field with BYTCHK set to one - checking protection information read from the medium (part 2 of 2)**

Code	Logical unit formatted with protection information	Field in protection information <sup>g</sup>	Extended INQUIRY Data VPD page bit value <sup>f</sup>	If check fails <sup>d e</sup> , additional sense code
<sup>a</sup> A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard. <sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND. <sup>e</sup> If multiple errors occur, then the selection of which error to report is not defined by this standard. <sup>f</sup> See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit. <sup>g</sup> If the device server detects a: a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.17.2.5) is enabled, then the device server shall not check any protection information in the associated logical block. <sup>h</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, then the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.				

If the BYTCHK bit is set to one, then the device server shall check the protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 69.

**Table 69 — VRPROTECT field with BYTCHK set to one - checking protection information from the data-out buffer (part 1 of 2)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d e</sup> , additional sense code
000b	Yes	No protection information received from application client to check		
	No	No protection information received from application client to check		
001b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		
010b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		
011b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition <sup>a</sup>		
100b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No	Error condition <sup>a</sup>		

**Table 69 — VRPROTECT field with BYTCHK set to one - checking protection information from the data-out buffer (part 2 of 2)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d e</sup> , additional sense code
101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>f</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			

<sup>a</sup> A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

<sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

<sup>c</sup> The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, then this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.

<sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

<sup>e</sup> If multiple errors occur, the selection of which error to report is not defined by this standard.

<sup>f</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server checks the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check the logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.



If the BYTCHK bit is set to one, then the device server shall perform a byte-by-byte comparison of protection information transferred from the data-out buffer with protection information read from the medium based on the VRPROTECT field as described in table 70.

**Table 70 — VRPROTECT field with BYTCHK set to one - byte-by-byte comparison requirements** (part 1 of 2)

Code	Logical unit formatted with protection information	Field	Byte-by-byte Comparison	If compare fails <sup>c d</sup> , additional sense code
000b	Yes	No protection information received from application client to compare. Only user data is compared within each logical block.		
	No	No protection information or the medium or received from application client to compare. Only user data is compared within each logical block.		
001b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) <sup>e</sup>	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) <sup>f</sup>	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG (not type 3)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)	Shall not	No compare performed
	No	Error condition <sup>a</sup>		
010b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall not	No compare performed
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) <sup>e</sup>	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) <sup>f</sup>	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG (not type 3)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)	Shall not	No compare performed
	No	Error condition <sup>a</sup>		

**Table 70 — VRPROTECT field with BYTCHK set to one - byte-by-byte comparison requirements** (part 2 of 2)

Code	Logical unit formatted with protection information	Field	Byte-by-byte Comparison	If compare fails <sup>c d</sup> , additional sense code
011b 100b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) <sup>e</sup>	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) <sup>f</sup>	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG (not type 3)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0)	Shall	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1)	Shall not	No compare performed
	No	Error condition <sup>a</sup>		
101b <sup>b</sup>	Yes	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 1) <sup>e</sup>	Shall	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG (ATO = 0) <sup>f</sup>	Shall not	No compare performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No compare performed
	No	Error condition <sup>a</sup>		
110b to 111b	Reserved			
<sup>a</sup> A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to MISCOMPARE. <sup>d</sup> If multiple errors occur, the selection of which error to report is not defined by this standard. <sup>e</sup> If the ATO bit is set to one in the Control mode page (see SPC-4), then the logical block application tag shall not be modified by a device server. <sup>f</sup> If the ATO bit is set to zero in the Control mode page (see SPC-4), then the logical block application tag may be modified by a device server.				

## 5.23 VERIFY (12) command

The VERIFY (12) command (see table 71) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

**Table 71 — VERIFY (12) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (AFh)							
1		VRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6	(MSB)	VERIFICATION LENGTH							
9									
10	Restricted for MMC-6	Reserved			GROUP NUMBER				
11		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 71.

See the VERIFY (10) command (see 5.22) for the definitions of the other fields in this command.

## 5.24 VERIFY (16) command

The VERIFY (16) command (see table 72) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

**Table 72 — VERIFY (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (8Fh)							
1		VRPROTECT			DPO	Reserved		BYTCHK	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10	(MSB)	VERIFICATION LENGTH							
13									
14	Restricted for MMC-6	Reserved			GROUP NUMBER				
15		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 72.

See the VERIFY (10) command (see 5.22) for the definitions of the other fields in this command.

## 5.25 VERIFY (32) command

The VERIFY (32) command (see table 73) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

The VERIFY (32) command shall only be processed if type 2 protection is enabled (see 4.17.2.4).

**Table 73 — VERIFY (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Ah)							
9									
10		VRPROTECT			DPO	Reserved		BYTCHK	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
23									
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG							
25									
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK							
27									
28	(MSB)	VERIFICATION LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 73.

See the VERIFY (10) command (see 5.22) for the definitions of the CONTROL byte, GROUP NUMBER field, VRPROTECT field, DPO bit, BYTCHK bit, LOGICAL BLOCK ADDRESS field, and VERIFICATION LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 67, table 68, table 69, and table 70 in 5.22), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 67, table 68, table 69, and table 70 in 5.22), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the

EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in the protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

- a) the ATO bit is set to zero; or
- b) the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 67, table 68, table 69, and table 70 in 5.22).

## 5.26 WRITE (6) command

The WRITE (6) command (see table 74) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data but does not include protection information. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

**Table 74 — WRITE (6) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (0Ah)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								
4	TRANSFER LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 74.

The cache control bits are not provided for this command. Direct-access block devices with cache may have values for the cache control bits that may affect the WRITE (6) command, however no default value is defined by this standard. If explicit control is required, then the WRITE (10) command should be used.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the data-out buffer and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be written. Any other value specifies the number of logical blocks that shall be written. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 20 - For the WRITE (10) command, WRITE (12) command, WRITE (16) command, and WRITE (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are transferred.

The contents of the CONTROL byte are defined in SAM-4.

If a WRITE (6) command is received after protection information is enabled, then the device server shall set the protection information (see 4.17) as follows as the device server writes each logical block to the medium:

- a) the LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.17.4);
- b) the LOGICAL BLOCK REFERENCE TAG field set to:
  - A) the least significant four bytes of the LBA, if type 1 protection (see 4.17.2.3) is enabled;
  - B) FFFF\_FFFFh, if type 2 protection is enabled (see 4.17.2.4);

- C) FFFF\_FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-4), and type 3 protection (see 4.17.2.5) is enabled; or
  - D) any value, if the ATO bit is set to zero in the Control mode page (see SPC-4), and type 3 protection (see 4.17.2.5) is enabled;
- and
- c) the LOGICAL BLOCK APPLICATION TAG field set to:
    - A) FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-4); or
    - B) any value, if the ATO bit is set to zero in the Control mode page (see SPC-4).

## 5.27 WRITE (10) command

The WRITE (10) command (see table 75) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

**Table 75 — WRITE (10) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB) LOGICAL BLOCK ADDRESS (LSB)							
5								
6	Reserved			GROUP NUMBER				
7	(MSB) TRANSFER LENGTH (LSB)							
8								
9	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 75.

See the READ (10) command (see 5.8) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

The contents of the CONTROL byte are defined in SAM-4.

The device server shall check the protection information transferred from the data-out buffer based on the WRPROTECT field as described in table 76.

**Table 76 — WRPROTECT field (part 1 of 3)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d i</sup> , additional sense code
000b	Yes <sup>f g h</sup>	No protection information received from application client to check		
	No	No protection information received from application client to check		

Table 76 — WRPROTECT field (part 2 of 3)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d i</sup> , additional sense code
001b <sup>b</sup>	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	Shall (except for type 3) <sup>j</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No <sup>a</sup>	No protection information available to check		
010b <sup>b</sup>	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>j</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No <sup>a</sup>	No protection information available to check		
011b <sup>b</sup>	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	Shall not	No check performed
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No <sup>a</sup>	No protection information available to check		
100b <sup>b</sup>	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	Shall not	No check performed
		LOGICAL BLOCK REFERENCE TAG	Shall not	No check performed
	No <sup>a</sup>	No protection information available to check		
101b <sup>b</sup>	Yes <sup>e</sup>	LOGICAL BLOCK GUARD	Shall	LOGICAL BLOCK GUARD CHECK FAILED
		LOGICAL BLOCK APPLICATION TAG	May <sup>c</sup>	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
		LOGICAL BLOCK REFERENCE TAG	May <sup>j</sup>	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
	No <sup>a</sup>	No protection information available to check		
110b to 111b	Reserved			

**Table 76 — WRPROTECT field (part 3 of 3)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails <sup>d i</sup> , additional sense code
<p><sup>a</sup> A write operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>b</sup> If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>c</sup> The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the WRITE (32) command (see 5.30) is used, then this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.</p> <p><sup>d</sup> If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.</p> <p><sup>e</sup> Device server shall preserve the contents of protection information (e.g., write to medium, store in non-volatile memory).</p> <p><sup>f</sup> The device server shall write a properly generated CRC (see 4.17.4.2) into each LOGICAL BLOCK GUARD field.</p> <p><sup>g</sup> If the P_TYPE field is set to 000h in the READ CAPACITY (16) parameter data (see 5.13), then the device server shall write the least significant four bytes of each LBA into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks. If the P_TYPE field is not set to 000b, then the device server shall write a value of FFFF_FFFFh into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.</p> <p><sup>h</sup> If the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, then the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.</p> <p><sup>i</sup> If multiple errors occur, the selection of which error to report is not defined by this standard.</p> <p><sup>j</sup> If type 1 protection is enabled, then the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server checks the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a WRITE (32) command (see 5.30). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check the logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.</p>				



The force unit access (FUA) and force unit access non-volatile cache (FUA\_NV) bits are defined in table 77.

**Table 77 — Force unit access for write operations**

FUA	FUA_NV	Description
0	0	The device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium.
0	1	If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall write the logical blocks to non-volatile cache and/or the medium.  If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium.
1	0 or 1	The device server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until the logical blocks have been written on the medium without error.

If logical blocks are transferred directly to a cache, then the device server may complete the command with GOOD status prior to writing the logical blocks to the medium. Any error that occurs after the device server has completed the command with GOOD status is returned as a deferred error, and information regarding the error is not reported until a subsequent command.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the data-out buffer and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be written. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be written. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

NOTE 21 - For the WRITE (6) command, a TRANSFER LENGTH field set to zero specifies that 256 logical blocks are transferred.

## 5.28 WRITE (12) command

The WRITE (12) command (see table 78) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

**Table 78 — WRITE (12) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (AAh)							
1		WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6	(MSB)	TRANSFER LENGTH							
9									
10	Restricted for MMC-6	Reserved			GROUP NUMBER				
11		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 78.

See the WRITE (10) command (see 5.27) for the definitions of the other fields in this command.

## 5.29 WRITE (16) command

The WRITE (16) command (see table 79) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

**Table 79 — WRITE (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (8Ah)							
1		WRPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10	(MSB)	TRANSFER LENGTH							
13									
14	Restricted for MMC-6	Reserved			GROUP NUMBER				
15		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 79.

See the WRITE (10) command (see 5.27) for the definitions of the other fields in this command.

### 5.30 WRITE (32) command

The WRITE (32) command (see table 80) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

The WRITE (32) command shall only be processed if type 2 protection is enabled (see 4.17.2.4).

**Table 80 — WRITE (32) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Bh)						(LSB)
9								
10	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Reserved
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						(LSB)
19								
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG						(LSB)
23								
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG						(LSB)
25								
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK						(LSB)
27								
28	(MSB)	TRANSFER LENGTH						(LSB)
31								

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 80.

See the WRITE (10) command (see 5.27) for the definitions of the CONTROL byte, GROUP NUMBER field, the WRPROTECT field, the DPO bit, the FUA bit, the FUA\_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 76 in 5.27), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field

expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 76 in 5.27), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in the protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

- a) the ATO bit is set to zero; or
- b) the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 76 in 5.27).

### 5.31 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 81) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium, and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

**Table 81 — WRITE AND VERIFY (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (2Eh)							
1		WRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 81.

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field. See the WRITE (10) command (see 5.27) for the definitions of the CONTROL byte, TRANSFER LENGTH field and the WRPROTECT field. See the READ (10) command (see 5.8) for the definition of the DPO bit.

If the Verify Error Recovery mode page (see 6.3.6) is also implemented, then the current settings in that mode page along with the AWRE bit in the Read-Write Error Recovery mode page (see 6.3.5) specify the verification error criteria. If these mode pages are not implemented, then the verification criteria is vendor-specific.

A byte check (BYTCHK) bit set to zero specifies that, after writing, the device server perform a medium verification with no data comparison. A BYTCHK bit set to one specifies that, after writing, the device server perform a byte-by-byte comparison of data written on the medium with the data just written. If the comparison is unsuccessful for any reason, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to the appropriate value for the condition.

### 5.32 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 82) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium, and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

**Table 82 — WRITE AND VERIFY (12) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (AEh)							
1		WRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6	(MSB)	TRANSFER LENGTH							
9									
10	Restricted for MMC-6	Reserved			GROUP NUMBER				
11		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 82.

See the WRITE AND VERIFY (10) command (see 5.31) for the definitions of the other fields in this command.

### 5.33 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 83) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium, and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

**Table 83 — WRITE AND VERIFY (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (8Eh)							
1		WRPROTECT			DPO	Reserved		BYTCHK	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
9									
10	(MSB)	TRANSFER LENGTH							
13									
14	Restricted for MMC-6	Reserved			GROUP NUMBER				
15		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 83.

See the WRITE AND VERIFY (10) command (see 5.31) for the definitions of the other fields in this command.

### 5.34 WRITE AND VERIFY (32) command

The WRITE AND VERIFY (32) command (see table 84) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium, and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

**Table 84 — WRITE AND VERIFY (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Ch)							
9									
10		WRPROTECT			DPO	Reserved		BYTCHK	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
23									
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG							
25									
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 84.

See the WRITE AND VERIFY (10) command (see 5.31) for the definitions of the CONTROL byte, GROUP NUMBER field, the WRPROTECT field, the DPO bit, the BYTCHK bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 76 in 5.27), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 76 in 5.27), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 76 in 5.27), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

### 5.35 WRITE LONG (10) command

The WRITE LONG (10) command (see table 85) requests that the device server mark a logical block or physical block as containing an error, or transfer data for a single logical block or physical block from the data-out buffer and write it to the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (10) command (see 5.16). The device server shall write the logical block or physical block to the medium, and shall not complete the command with GOOD status until the logical block has been written on the medium without error.

**Table 85 — WRITE LONG (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (3Fh)							
1		COR_DIS	WR_UNCOR	PBLOCK	Reserved				Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved							
7	(MSB)	BYTE TRANSFER LENGTH							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 85.

The correction disabled (COR\_DIS) bit, the write uncorrectable error (WR\_UNCOR) bit, and the physical block (PBLOCK) bit are defined in table 86. If there is more than one logical block per physical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) data (see 5.13.1) is set to a non-zero value), then the device server shall support the WR\_UNCOR bit and the PBLOCK bit.

**Table 86 — COR\_DIS bit, WR\_UNCOR bit, and PBLOCK bit (part 1 of 2)**

COR_DIS	WR_UNCOR	PBLOCK	More than one logical block per physical block <sup>a</sup>	Description
0	0	0	yes or no	Write only the specified logical block using the value in the BYTE TRANSFER LENGTH field.
		1	no	Complete the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
			yes	Write the entire physical block containing the specified logical block using the value in the BYTE TRANSFER LENGTH field.
0	1	0	yes or no	Mark only the specified logical block as containing a pseudo unrecovered error with correction enabled (see 4.14.2) in a manner that causes the device server to perform the maximum error recovery as defined by the Read-Write Error Recovery mode page (see 6.3.5).  Ignore the BYTE TRANSFER LENGTH field, and transfer no data.
			no	Complete the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
		1	yes	Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction enabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction enabled) (see 4.14.2).  Ignore the BYTE TRANSFER LENGTH field, and transfer no data.

<sup>a</sup> An entry of “yes” means that the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) data (see 5.13.1) is set to a non-zero value. An entry of “no” means that the field is set to zero.



Table 86 — COR\_DIS bit, WR\_UNCOR bit, and PBLOCK bit (part 2 of 2)

COR_DIS	WR_UNCOR	PBLOCK	More than one logical block per physical block <sup>a</sup>	Description
1	0	0	yes or no	Mark only the specified logical block as containing a pseudo unrecovered error with correction disabled (see 4.14.2)  Write only the specified logical block using the value in the BYTE TRANSFER LENGTH field.
		1	no	Complete the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
			yes	Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction disabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction disabled) (see 4.14.2).  Write the entire physical block containing the specified logical block using the value in the BYTE TRANSFER LENGTH field.
1	1	0	yes or no	Mark only the specified logical block as containing a pseudo unrecovered error with correction disabled (see 4.14.2).  Ignore the BYTE TRANSFER LENGTH field, and transfer no data.
		1	no	Complete the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
			yes	Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction disabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction disabled) (see 4.14.2).  Ignore the BYTE TRANSFER LENGTH field, and transfer no data.

<sup>a</sup> An entry of “yes” means that the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) data (see 5.13.1) is set to a non-zero value. An entry of “no” means that the field is set to zero.

Any pseudo unrecovered error with correction disabled shall remain in effect until the logical block is written by any means (e.g., another WRITE LONG command with the COR\_DIS bit set to zero and the WR\_UNCOR bit

set to zero that writes to the same logical block, any WRITE command that specifies the writes to the same logical block, or a FORMAT UNIT command).

In the Extended INQUIRY Data VPD page (see SPC-4), the setting of the CD\_SUP bit indicates whether or not the logical unit supports the CD\_SUP bit being set to one, and the setting of the WU\_SUP bit indicates whether or not the logical unit supports the WR\_UNCOR bit being set to one.

The LOGICAL BLOCK ADDRESS field specifies an LBA. If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If table 86 defines that the value in the BYTE TRANSFER LENGTH field is used, then the BYTE TRANSFER LENGTH field specifies the number of bytes of data that the device server shall transfer from the data-out buffer and write to the specified logical block or physical block. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the data length that the device server returns for a READ LONG command, then the device server shall complete the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.14 and SPC-4), the ILI and VALID bits shall be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested length minus the actual length in bytes. Negative values shall be indicated by two's complement notation. If the BYTE TRANSFER LENGTH field is set to zero, then no bytes shall be written. This condition shall not be considered an error.

The contents of the CONTROL byte are defined in SAM-4.

### 5.36 WRITE LONG (16) command

The WRITE LONG (16) command (see table 87) requests that the device server mark a logical block or physical block as containing an error, or transfer data for a single logical block or physical block from the data-out buffer and write it to the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (16) command (see 5.17). The device server shall write the logical block or physical block to the medium, and shall not complete the command with GOOD status until the logical block has been written on the medium without error. This command is implemented as a service action of the SERVICE ACTION OUT operation code (see A.2).

**Table 87 — WRITE LONG (16) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (9Fh)							
1		COR_DIS	WR_UNCOR	PBLOCK	SERVICE ACTION (11h)				
2		(MSB)							
9		LOGICAL BLOCK ADDRESS							
		(LSB)							
10		Reserved							
11									
12		(MSB)							
		BYTE TRANSFER LENGTH							
13		(LSB)							
14		Reserved							
15		CONTROL							

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 87.

See the WRITE LONG (10) command (see 5.35) for the definitions of the fields in this command.

### 5.37 WRITE SAME (10) command

The WRITE SAME (10) command (see table 88) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

**Table 88 — WRITE SAME (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (41h)							
1		WRPROTECT			Reserved		PBDATA	LBDATA	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	NUMBER OF LOGICAL BLOCKS							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 88.

See the WRITE (10) command (see 5.27) for the definitions of the CONTROL byte and WRPROTECT field. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

Table 89 describes the LBDATA bit and the PBDATA bit.

**Table 89 — LBDATA bit and PBDATA bit**

LBDATA	PBDATA	Description
0	0	<p>The device server shall write the single block of user data received from the data-out buffer to each logical block without modification.</p> <p>If the medium is formatted with type 1 or type 2 protection information, then:</p> <ul style="list-style-type: none"> <li>a) the value in the LOGICAL BLOCK REFERENCE TAG field received in the single block of data from the data-out buffer shall be placed into the LOGICAL BLOCK REFERENCE TAG field of the first logical block written to the medium. Into each of the subsequent logical blocks, the device server shall place into the LOGICAL BLOCK REFERENCE TAG field the value of the previous logical block's LOGICAL BLOCK REFERENCE TAG field plus one;</li> <li>b) If the ATO bit is set to one in the Control mode page (see SPC-4), then the logical block application tag received in the single block of data shall be placed in the LOGICAL BLOCK APPLICATION TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK APPLICATION TAG field of each logical block; and</li> <li>c) The value in the LOGICAL BLOCK GUARD field received in the single block of data from the data-out buffer shall be placed in the LOGICAL BLOCK GUARD field of each logical block.</li> </ul> <p>If the medium is formatted with type 3 protection information:</p> <ul style="list-style-type: none"> <li>a) If the ATO bit is set to one in the Control mode page (see SPC-4), then the logical block reference tag received in the single block of data shall be placed in the LOGICAL BLOCK REFERENCE TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field of each logical block;</li> <li>b) If the ATO bit is set to one in the Control mode page (see SPC-4), then the logical block application tag received in the single block of data shall be placed in the LOGICAL BLOCK REFERENCE TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field of each logical block; and</li> <li>c) The value in the LOGICAL BLOCK GUARD field received in the single block of data from the data-out buffer shall be placed in the LOGICAL BLOCK GUARD field of each logical block.</li> </ul>
0	1 <sup>a</sup>	The device server shall replace the first eight bytes of the block received from the data-out buffer to each physical sector with the physical address of the sector being written using the physical sector format (see 5.2.2.4.5).
1 <sup>a</sup>	0	The device server shall replace the first four bytes of the block received from the data-out buffer with the least significant four bytes of the LBA of the block being written, ending with the least significant byte (e.g., if the LBA is 7766_5544_3322_1100h, 3322_1100h is written with 33h written first and 00h written last).
1	1	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
<sup>a</sup> If the medium is formatted with protection information, then the protection information shall be written to a default value of FFFF_FFFF_FFFF_FFFFh in each of the written logical blocks.		

The NUMBER OF LOGICAL BLOCKS field specifies the number of contiguous logical blocks to be written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to

zero specifies that the device server write all the logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium. If the LBA plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

### 5.38 WRITE SAME (16) command

The WRITE SAME (16) command (see table 90) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

**Table 90 — WRITE SAME (16) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (93h)							
1	WRPROTECT			Reserved		PBDATA	LBDATA	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	NUMBER OF LOGICAL BLOCKS							(LSB)
14	Reserved			GROUP NUMBER				
15	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 90.

See the WRITE SAME (10) command (see 5.37) for the definitions of the other fields in this command.

### 5.39 WRITE SAME (32) command

The WRITE SAME (32) command (see table 91) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

**Table 91 — WRITE SAME (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (000Dh)							
9									
10		WRPROTECT			Reserved		PBDATA	LBDATA	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20	(MSB)	EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG							
23									
24	(MSB)	EXPECTED LOGICAL BLOCK APPLICATION TAG							
25									
26	(MSB)	LOGICAL BLOCK APPLICATION TAG MASK							
27									
28	(MSB)	NUMBER OF LOGICAL BLOCKS							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 91.

See the WRITE SAME (10) command (see 5.37) for the definitions of the CONTROL byte, GROUP NUMBER field, the WRPROTECT field, the PBDATA bit, the LBDATA bit, the LOGICAL BLOCK ADDRESS field, and the NUMBER OF LOGICAL BLOCKS field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 76 in 5.27), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 76 in 5.27), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG

MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 76 in 5.27), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

## 5.40 XDREAD (10) command

The XDREAD (10) command (see table 92) requests that the device transfer to the data-in buffer the XOR data generated by an XDWRITE command (see 5.42 and 5.43). XOR data includes user data and may include protection information, based on the XORPINFO bit and the medium format.

**Table 92 — XDREAD (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (52h)							
1		Reserved							XORPINFO
2	(MSB)	LOGICAL BLOCK ADDRESS							
5		(LSB)							
6		Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH							
8		(LSB)							
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 92.

See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

If the XOR protection information (XORPINFO) bit is set to zero, then the device server shall not check or transmit protection information.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has been formatted with protection information, then the device server shall transmit protection information but shall not check any of the protection information fields.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the XORPINFO bit is set to one, and the device server does not support protection information, then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The XOR data transferred is identified by the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field. The LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field shall be the same as, or a subset of, those specified in a prior XDWRITE command. If a match is not found, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The contents of the CONTROL byte are defined in SAM-4.

## 5.41 XDREAD (32) command

The XDREAD (32) command (see table 93) requests that the device transfer to the data-in buffer the XOR data generated by an XDWRITE command (see 5.42 and 5.43). XOR data includes user data and may include protection information, based on the XORPINFO bit and the medium format.

**Table 93 — XDREAD (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0003h)							
9									
10		Reserved							XORPINFO
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20		Reserved							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the value defined in table 93.

See the XDREAD (10) command (see 5.40) and SPC-4 for the definitions of the other fields in this command.



## 5.42 XDWRITE (10) command

The XDWRITE (10) command (see table 94) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer;
- 3) perform an XOR operation with the logical blocks transferred from the data-out buffer and the logical blocks read, storing the resulting XOR data in a buffer; and
- 4) if the DISABLE WRITE bit is set to zero, then write the logical blocks transferred from the data-out buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The resulting XOR data shall be retained as specified in 4.15.4.

**Table 94 — XDWRITE (10) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (50h)							
1		WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	Reserved
2	(MSB)	LOGICAL BLOCK ADDRESS							
5									
6		Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH							
8									
9		CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 94.

See the WRITE (10) command (see 5.27) for the definitions of the WRPROTECT field, the FUA bit, and the FUA\_NV bit. See the READ (10) command (see 5.8) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

A DISABLE WRITE bit set to zero specifies that the data transferred from the data-out buffer shall be written to the medium after the XOR operation is complete. A DISABLE WRITE bit set to one specifies that the data shall not be written to the medium.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that read, transferred from the data-out buffer, and XORed into a buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The resulting XOR data is retrieved by an XDREAD command (see 5.40 and 5.41) with starting LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field that match, or are a subset of, the LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field of this command.

The contents of the CONTROL byte are defined in SAM-4.

### 5.43 XDWRITE (32) command

The XDWRITE (32) command (see table 95) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer;
- 3) perform an XOR operation with the logical blocks transferred from the data-out buffer and the logical blocks read, storing the resulting XOR data in a buffer; and
- 4) if the DISABLE WRITE bit is set to zero, then write the logical blocks transferred from the data-out buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The resulting XOR data shall be retained as specified in 4.15.4.

**Table 95 — XDWRITE (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0004h)							
9									
10		WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	Reserved
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20		Reserved							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 95.

See the XDWRITE (10) command (see 5.42) and SPC-4 for the definitions of the other fields in this command.

## 5.44 XDWRITEREAD (10) command

The XDWRITEREAD (10) command (see table 96) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer;
- 3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, storing the resulting XOR data in a buffer;
- 4) if the DISABLE WRITE bit is set to zero, then write the logical blocks transferred from the data-out buffer; and
- 5) transfer the resulting XOR data to the data-in buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field, the XORPINFO bit, and the medium format. This command is equivalent to an XDWRITE (10) command (see 5.42) followed by an XDREAD (10) command (see 5.40) specifying the same values for the GROUP NUMBER field, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field. This command is only available on transport protocols supporting bidirectional commands.

**Table 96 — XDWRITEREAD (10) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (53h)							
1	WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	XORPINFO
2	(MSB) _____							
5	LOGICAL BLOCK ADDRESS _____ (LSB)							
6	Reserved			GROUP NUMBER				
7	(MSB) _____							
8	TRANSFER LENGTH _____ (LSB)							
9	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 96.

See the XDWRITE (10) command (see 5.42) and XDREAD (10) command (see 5.40) for the definitions of the other fields in this command.

## 5.45 XDWRITEREAD (32) command

The XDWRITEREAD (32) command (see table 97) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer;
- 3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, storing the resulting XOR data in a buffer;
- 4) if the DISABLE WRITE bit is set to zero, then write the logical blocks transferred from the data-out buffer; and
- 5) transfer the resulting XOR data to the data-in buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field, the XORPINFO bit, and the medium format. This command is equivalent to an XDWRITE (32) command (see 5.43) followed by an XDREAD (32) command (see 5.41) specifying the same values for the GROUP NUMBER field, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field. This command is only available on transport protocols supporting bidirectional commands.

**Table 97 — XDWRITEREAD (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0007h)							
9									
10		WRPROTECT			DPO	FUA	DISABLE WRITE	FUA_NV	XORPINFO
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20		Reserved							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 97.

See the XDWRITEREAD (10) command (see 5.44) and SPC-4 for the definitions of the fields in this command.

## 5.46 XPWRITE (10) command

The XPWRITE (10) command (see table 98) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer; and
- 3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, writing the resulting XOR data.

Each logical block includes user data and may include protection information, based on the XORPINFO bit and the medium format.

**Table 98 — XPWRITE (10) command**

Bit	7	6	5	4	3	2	1	0
Byte								
0	OPERATION CODE (51h)							
1	Reserved			DPO	FUA	Reserved	FUA_NV	XORPINFO
2	(MSB) LOGICAL BLOCK ADDRESS (LSB)							
5								
6	Reserved			GROUP NUMBER				
7	(MSB) TRANSFER LENGTH (LSB)							
8								
9	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 98.

See the READ (10) command (see 5.8) for the definition of the DPO bit. See the WRITE (10) command (see 5.27) for the definitions of the FUA bit and the FUA\_NV bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

If the XOR protection information (XORPINFO) bit is set to zero, the device server supports protection information, and the medium has been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has been formatted with protection information, then the device server shall XOR the user data and protection information transferred from the data-out buffer with the user data and protection information read, and then write the resulting XOR data. The device server shall not check any of the protection information fields.

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the XORPINFO bit is set to one, and the device server does not support protection information, then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks that shall be read, XORed with logical blocks transferred from the data-out buffer, and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the LBA plus the transfer length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

The contents of the CONTROL byte are defined in SAM-4.

### 5.47 XPWRITE (32) command

The XPWRITE (32) command (see table 99) requests that the device server:

- 1) read the specified logical block(s);
- 2) transfer logical blocks from the data-out buffer; and
- 3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, writing the resulting XOR data.

Each logical block includes user data and may include protection information, based on the XORPINFO bit and the medium format.

**Table 99 — XPWRITE (32) command**

Byte	Bit	7	6	5	4	3	2	1	0
0		OPERATION CODE (7Fh)							
1		CONTROL							
2		Reserved							
5									
6		Reserved			GROUP NUMBER				
7		ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0006h)							
9									
10		Reserved			DPO	FUA	Reserved	FUA_NV	XORPINFO
11		Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS							
19									
20		Reserved							
27									
28	(MSB)	TRANSFER LENGTH							
31									

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 99.

See the XPWRITE (10) command (see 5.46) and SPC-4 for the definitions of the fields in this command.

## 6 Parameters for direct-access block devices

### 6.1 Diagnostic parameters

#### 6.1.1 Diagnostic parameters overview

This subclause defines the descriptors and pages for diagnostic parameters used with direct-access block devices. The diagnostic page codes for direct-access block devices are defined in table 100.

**Table 100 — Diagnostic page codes**

Diagnostic page code	Description	Reference
00h	Supported diagnostic pages	SPC-4
01h to 2Fh	SCSI enclosure services diagnostic pages	SES-2
30h to 3Fh	Diagnostic pages assigned by SPC-4	SPC-4
40h	Translate Address Output diagnostic page	6.1.2
	Translate Address Input diagnostic page	6.1.3
41h	Obsolete (Device Status diagnostic pages)	
42h to 7Fh	Reserved for this standard	
80h to FFh	Vendor-specific diagnostic pages	

#### 6.1.2 Translate Address Output diagnostic page

The Translate Address diagnostic pages allow the application client to translate an address in one of the formats supported by the FORMAT UNIT command (see 5.2.2.4) (i.e., a short block format address, a long block format address, a physical sector format address, or a bytes from index format address) into any one of the other formats. The address to be translated is sent to the device server with the SEND DIAGNOSTIC command and the results are returned to the application client by the RECEIVE DIAGNOSTIC RESULTS command.

Table 101 defines the format of the Translate Address Output diagnostic page sent with the SEND DIAGNOSTIC command. The translated address is returned in the Translate Address Input diagnostic page (see 6.1.3).

**Table 101 — Translate Address Output diagnostic page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PAGE CODE (40h)							
1		Reserved							
2	(MSB)	PAGE LENGTH (000Ah)							
3		(LSB)							
4		Reserved					SUPPLIED FORMAT		
5		Reserved					TRANSLATE FORMAT		
6	(MSB)	ADDRESS TO TRANSLATE							
13		(LSB)							

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 101.

The SUPPLIED FORMAT field specifies the format of the ADDRESS TO TRANSLATE field. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command (see 5.2). If the device server does not support the requested format it shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The TRANSLATE FORMAT field specifies the format the device server shall use for the result of the address translation. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command. If the device server does not support the specified format it shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ADDRESS TO TRANSLATE field contains a single address descriptor which the application client is requesting the device server to translate. The format of this field depends on the value in the SUPPLIED FORMAT field. The formats are described in 5.2.2.4. If the short block format address descriptor is specified, the first four bytes of the ADDRESS TO TRANSLATE field shall contain the short block format address descriptor and the last four bytes shall contain 0000\_0000h.

### 6.1.3 Translate Address Input diagnostic page

Table 102 defines the Translate Address Input diagnostic page retrieved with the RECEIVE DIAGNOSTIC RESULTS command after the Translate Address Output diagnostic page (see 6.1.2) has been sent with the SEND DIAGNOSTIC command. If a Translate Address Output diagnostic page has not yet been processed, the results of a RECEIVE DIAGNOSTIC RESULTS command requesting this diagnostic page are vendor-specific.

**Table 102 — Translate Address Input diagnostic page**

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n - 3)						(LSB)
3								
4	Reserved					SUPPLIED FORMAT		
5	RAREA	ALTSEC	ALTTRK	Reserved		TRANSLATED FORMAT		
Translated address(es)								
6	(MSB)	TRANSLATED ADDRESS 1						(LSB)
13								
	...							
n - 7	(MSB)	TRANSLATED ADDRESS x (if required)						(LSB)
n								

The PAGE CODE field is defined in SPC-4 and shall be set to the value defined in table 102.

The PAGE LENGTH field is defined in SPC-4.

The Translate Address Input diagnostic page contains a four-byte page header that specifies the page code and length followed by two bytes that describe the translated address followed by zero or more translated addresses.

The PAGE LENGTH field contains the number of parameter bytes that follow.



The SUPPLIED FORMAT field contains the value from the SUPPLIED FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

A reserved area (RAREA) bit set to zero indicates that no part of the translated address falls within a reserved area of the medium. A RAREA bit set to one indicates that all or part of the translated address falls within a reserved area of the medium (e.g., speed tolerance gap, alternate sector, or vendor reserved area). If the entire translated address falls within a reserved area, the device server may not return a translated address.

An alternate sector (ALTSEC) bit set to zero indicates that no part of the translated address is located in an alternate sector of the medium or that the device server is unable to determine this information. An ALTSEC bit set to one indicates that the translated address is physically located in an alternate sector of the medium. If the device server is unable to determine if all or part of the translated address is located in an alternate sector it shall set this bit to zero.

An alternate track (ALTTRK) bit set to zero indicates that no part of the translated address is located on an alternate track of the medium. An ALTTRK bit set to one indicates that part or all of the translated address is located on an alternate track of the medium or the device server is unable to determine if all or part of the translated address is located on an alternate track.

The TRANSLATED FORMAT field contains the value from the TRANSLATE FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

The TRANSLATED ADDRESS field(s) contains the address(es) the device server translated from the address supplied by the application client in the previous Translate Address Output diagnostic page. Each field shall be in the format specified in the TRANSLATE FORMAT field. The formats are described in 5.2.2.4. If the short block format address descriptor is specified, the first four bytes of the TRANSLATED ADDRESS field shall contain the short block format address descriptor and the last four bytes shall contain 0000\_0000h.

If the returned data is in short block format, long block format, or physical sector format and the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page covers more than one address after it has been translated (e.g., because of multiple physical sectors within a single logical block or multiple logical blocks within a single physical sector) the device server shall return all possible addresses that are contained in the area specified by the address to be translated. If the returned data is in bytes from index format, the device server shall return a pair of translated values for each of the possible addresses that are contained in the area specified by the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page. Of the pair of translated values returned, the first indicates the starting location and the second the ending location of the area.

## 6.2 Log parameters

### 6.2.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with direct-access block devices. See SPC-4 for a detailed description of logging operations. The log page codes for direct-access block devices are defined in table 103.

**Table 103 — Log page codes**

Page code <sup>a</sup>	Subpage code <sup>a</sup>	Description	Reference
00h	00h	Supported Log Pages log page	SPC-4
00h	FFh	Supported Log Pages and Subpages	SPC-4
01h	00h	Buffer Over-Run/Under-Run log page	SPC-4
01h to 3Fh	FFh	Supported Subpages	SPC-4
02h	00h	Write Error Counter log page	SPC-4
03h	00h	Read Error Counter log page	SPC-4
04h	00h to FEh	Reserved	
05h	00h	Verify Error Counter log page	SPC-4
06h	00h	Non-Medium Error log page	SPC-4
07h	00h	Last n Error Events log page	SPC-4
08h	00h	Format Status log page	6.2.3
09h	00h	Restricted (see SPC-4)	
0Ah	00h	Restricted (see SPC-4)	
0Bh	00h	Last n Deferred Errors Or Asynchronous Events log page	SPC-4
0Ch	00h	Reserved	
0Dh	00h	Temperature log page	SPC-4
0Eh	00h	Start-Stop Cycle Counter log page	SPC-4
0Fh	00h	Application Client log page	SPC-4
10h	00h	Self-Test Results log page	SPC-4
11h to 14h	00h to FEh	Reserved	
15h	00h	Background Scan Results log page	6.2.2
16h	00h to FEh	Reserved	
17h	00h	Non-volatile Cache log page	6.2.4
18h	00h to FEh	Protocol-Specific Port log pages	SPC-4
19h to 2Eh	00h to FEh	Reserved	
2Fh	00h	Informational Exceptions log page	SPC-4
30h to 3Eh	00h to FEh	Vendor-specific	
3Fh	00h to FEh	Reserved	
<sup>a</sup> All page code and subpage code combinations not shown in this table are reserved.			

### 6.2.2 Background Scan Results log page

The Background Scan Results log page (see table 104) contains the Background Scanning Status parameter and zero or more Background Medium Scan parameters. The Background Scanning Status parameter provides information about background pre-scan and background medium scan operations. Each Background Medium Scan parameter provides information about a logical block where an error was detected during a background scanning operation. If the Background Scan Results log page is full, and a new error is detected during a background scanning operation, then the device server overwrites the oldest Background Medium Scan parameter with the Background Medium Scan parameter for the new error. When a LOG SELECT command with the PCR bit set to one is processed, the device server shall:

- a) delete all Background Medium Scan parameters; and
- b) not change the values in the Background Scanning Status parameter.

**Table 104 — Background Scan Results log page**

Byte	Bit	7	6	5	4	3	2	1	0
0		DS (1)	SPF (0)	PAGE CODE (15h)					
1		SUBPAGE CODE (00h)							
2		(MSB)	PAGE LENGTH (n - 3)						
3									
Background Scan Results log parameters									
4		Background Scanning Status parameter (see table 106)							
19									
Background Medium Scan parameter list									
20		Background Medium Scan parameter (first) (see table 109)							
43									
		.....							
n-23		Background Medium Scan parameter (last) (see table 109)							
n									

The disable save (DS) bit, the SubPage Format (SPF) bit, the PAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The disable save (DS) bit, the SubPage Format (SPF) bit, and the PAGE CODE field shall be set to the values defined in table 104.

Table 105 defines the parameter codes for the Background Scan Results log page.

**Table 105 — Background Scan Results log page parameter codes**

Parameter code	Description
0000h	Background Scanning Status
0001h to 0800h	Background Medium Scan
0801h to FFFFh	Reserved

The Background Scanning Status parameter (see table 106) contains status information about the background pre-scan and background medium scan features.

**Table 106 — Background Scanning Status parameter format**

Byte	Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)							
1		(LSB)							
2	Parameter control byte								
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING		
3		PARAMETER LENGTH (0Ch)							
4	(MSB)	ACCUMULATED POWER ON MINUTES							
7		(LSB)							
8		Reserved							
9		BACKGROUND SCANNING STATUS							
10	(MSB)	NUMBER OF BACKGROUND SCANS PERFORMED							
11		(LSB)							
12	(MSB)	BACKGROUND MEDIUM SCAN PROGRESS							
13		(LSB)							
14	(MSB)	NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED							
15		(LSB)							

The PARAMETER CODE field shall be set to the value defined in table 106.

Table 107 defines the values for the log parameter control byte (see SPC-4) for this log parameter.

**Table 107 — Parameter control byte for Background Scanning Status log parameter**

Field	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0	0 or 1 (i.e., ignored)	The DU bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0	0 or 1 (i.e., ignored)	The ETC bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TMC	00b	any (i.e., ignored)	The TMC field is not defined for list parameters, so shall be set to 00b when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
FORMAT AND LINKING	11b	11b	The log parameter is a binary format list parameter.

The PARAMETER LENGTH field indicates the number of bytes to follow in the log parameter and shall be set to the value defined in table 106.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing.

Table 108 defines the BACKGROUND SCANNING STATUS field.

**Table 108 — BACKGROUND SCANNING STATUS field**

Code	Description
00h	No background scans active, and background scanning is disabled (i.e., the EN_BMS bit is set to zero in the Background Control mode page (see 6.3.3)).
01h	Background medium scan is active
02h	Background pre-scan is active
03h	Background medium scan halted due to fatal error
04h	Background medium scan halted due to a vendor-specific pattern of errors
05h	Background medium scan halted due to medium formatted without P-list
06h	Background medium scan halted - vendor-specific cause
07h	Background medium scan halted due to temperature out of allowed range
08h	Background scanning is enabled (i.e., the EN_BMS bit is set to one in the Background Control mode page (see 6.3.3)), and no background medium scan is active (i.e., the device server is waiting for Background Medium Scan Interval timer expiration before starting the next background medium scan).
09h to FFh	Reserved

The NUMBER OF BACKGROUND SCANS PERFORMED field indicates the number of background scans (i.e., the total number of background pre-scans plus the number of background medium scans) that have been performed since the SCSI target device was originally shipped by the manufacturer.

The BACKGROUND MEDIUM SCAN PROGRESS field is a percent complete indication of the background medium scan. The returned value is a numerator that has 65 536 (i.e., 1\_0000h) as its denominator. If there is no background scan in progress (i.e., no background scan has been initiated since power on or the most recent scan has completed), then the device server shall set the BACKGROUND MEDIUM SCAN PROGRESS field to 0000h.

The NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field indicates the number of background medium scans that have been performed since the SCSI target device was originally shipped by the manufacturer. If the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field contains 0000h, then the number of background medium scans is not reported.

The total number of background pre-scans that have been performed is the value in the NUMBER OF BACKGROUND SCANS PERFORMED field minus the value in the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field.

A Background Medium Scan parameter (see table 109) describes a defect location on the medium that was encountered during a background scanning operation (see 4.19.2 and 4.19.3).

**Table 109 — Background Medium Scan parameter format**

Byte	Bit	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h to 0800h)							
1									
2	Parameter control byte								
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING		
3	PARAMETER LENGTH (14h)								
4	(MSB)	ACCUMULATED POWER ON MINUTES							
7									
8	REASSIGN STATUS				SENSE KEY				
9	ADDITIONAL SENSE CODE								
10	ADDITIONAL SENSE CODE QUALIFIER								
11	Vendor-specific								
15									
16	(MSB)	LOGICAL BLOCK ADDRESS							
23									

Table 110 defines the values for the log parameter control byte (see SPC-4) for this log parameter.

**Table 110 — Parameter control byte for Background Medium Scan log parameter**

Field	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0	0 or 1 (i.e., ignored)	The DU bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0	0 or 1 (i.e., ignored)	The ETC bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TMC	00b	any (i.e., ignored)	The TMC field is not defined for list parameters, so shall be set to 00b when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
FORMAT AND LINKING	11b	11b	The log parameter is a binary format list parameter.

The PARAMETER LENGTH field indicates the number of bytes to follow in the log parameter shall be set to the value defined in table 109.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing at the time the background scan error occurred.

Table 111 defines the REASSIGN STATUS field.

**Table 111 — REASSIGN STATUS field** (part 1 of 2)

Code	Reported	Description
0h	No	Reserved
1h	Yes	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation, and reassignment the logical block is pending receipt of: <sup>a b</sup> a) a command performing a write operation, if automatic write reallocation is allowed (i.e., the AWRE bit is set to one in the Read-Write Error Recovery mode page (see 6.3.6); or b) a REASSIGN BLOCKS command (see 5.18).
2h	No	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation and the logical block was successfully reassigned by the device server with recovered data.
3h	Reserved	
4h	Yes	An error was detected: a) while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation; b) reassignment of the logical block by the device server failed; and c) the logical block may or may not have an uncorrectable error.
5h	No	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation and the error was corrected by the device server rewriting the logical block without reassignment.
6h	Yes	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation, and the logical block: a) was successfully reassigned by the application client; and b) contains valid data (e.g., as the result of reassignment by a REASSIGN BLOCKS command during which the data was recovered, or by a command performing a write operation). <sup>b</sup>
7h	Yes	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation, and the logical block: a) was successfully reassigned by the application client; and b) does not contain valid data (e.g., as a result of reassignment by a REASSIGN BLOCKS command during which the data was not recovered). <sup>b</sup>
<p><b>Key:</b></p> <p>Yes = The LOWIR bit is set to one in the Background Control mode page (see 6.3.3).  No = The LOWIR bit is set to zero in the Background Control mode page.</p> <p><sup>a</sup> If the application client knows the correct data for the logical block, then the application client should use a command performing a write operation to reassign the logical block using the correct data (e.g., in a redundancy group (see 4.15.1), the application client writes data to the logical block regenerated from the data on the other logical units in the redundancy group). If the application client uses a REASSIGN BLOCKS command to reassign the logical block, then the device server may not be able to recover the data and does not report whether or not the data was recovered.</p> <p><sup>b</sup> The REASSIGN STATUS field in a given log parameter changes from 1h or 4h to 6h, 7h, or 8h when the logical block is reassigned, rewritten, or failed. If the logical block is reassigned or rewritten, any subsequent medium error occurring for the logical block is reported in a new log parameter with the same value in the LOGICAL BLOCK ADDRESS field as the value in the LOGICAL BLOCK ADDRESS field in the log parameter for the previous medium error for the logical block.</p>		



**Table 111 — REASSIGN STATUS field (part 2 of 2)**

Code	Reported	Description
8h	Yes	An error was detected while reading the logical block specified by the LOGICAL BLOCK ADDRESS field during a background scanning operation and the logical block was not successfully reassigned by the application client (e.g., by a REASSIGN BLOCKS command that failed)
9h to Fh	Reserved	
Key:		
Yes = The LOWIR bit is set to one in the Background Control mode page (see 6.3.3).		
No = The LOWIR bit is set to zero in the Background Control mode page.		
<sup>a</sup> If the application client knows the correct data for the logical block, then the application client should use a command performing a write operation to reassign the logical block using the correct data (e.g., in a redundancy group (see 4.15.1), the application client writes data to the logical block regenerated from the data on the other logical units in the redundancy group). If the application client uses a REASSIGN BLOCKS command to reassign the logical block, then the device server may not be able to recover the data and does not report whether or not the data was recovered. <sup>b</sup> The REASSIGN STATUS field in a given log parameter changes from 1h or 4h to 6h, 7h, or 8h when the logical block is reassigned, rewritten, or failed. If the logical block is reassigned or rewritten, any subsequent medium error occurring for the logical block is reported in a new log parameter with the same value in the LOGICAL BLOCK ADDRESS field as the value in the LOGICAL BLOCK ADDRESS field in the log parameter for the previous medium error for the logical block.		

The SENSE KEY field, ADDITIONAL SENSE CODE field, and the ADDITIONAL SENSE CODE QUALIFIER field may contain a hierarchy of additional information relating to error conditions that occurred during background scanning. The content of these fields is represented in the same format used by the sense data (see SPC-4).

The LOGICAL BLOCK ADDRESS field indicates the LBA associated with the medium error.

### 6.2.3 Format Status log page

The Format Status log page (log page code 08h) captures the state of the direct-access block device since the most recent successful FORMAT UNIT command (see 5.2) was completed. Additionally, this log page provides Defect Management information for the device server.

The Format Status log page uses the log page format defined in SPC-4.

Table 112 defines the parameter codes for the Format Status log page.

**Table 112 — Format Status log page parameter codes**

Parameter code	Description
0000h	Format Data Out
0001h	Grown Defects During Certification
0002h	Total Blocks Reallocated During Format
0003h	Total New Blocks Reallocated
0004h	Power On Minutes Since Format
0005h to 7FFFh	Reserved
8000h to FFFFh	Vendor-specific

The PARAMETER LENGTH field of each log parameter (see SPC-4) contains the length of the corresponding PARAMETER VALUE field and is vendor-specific.

Event counts are returned as a result of the LOG SENSE command. The default value for each event count listed table 112 shall be zero. Attempts to change these event counts by issuing a LOG SELECT with these fields set to non-zero values is not considered an error and shall have no effect on the saved values.

If information about a log parameter is not available, the device server shall return a value with each byte set to FFh (e.g., if the PARAMETER LENGTH field is set to 02h, the PARAMETER VALUE field is set to FFFFh). If the most recent FORMAT UNIT command failed, the device server shall return a value with each byte set to FFh for each log parameter.

The Format Data Out parameter contains the entire FORMAT UNIT parameter list (see 5.2.2) from the most recent successful FORMAT UNIT command. This includes:

- a) the parameter list header;
- b) the initialization pattern descriptor, if any; and
- c) the defect list, if any.

The Grown Defects During Certification parameter is a count of the number of defects detected as a result of performing certification during processing of the most recent successful FORMAT UNIT command. This count reflects only those defects detected and replaced that were not already part of the PLIST or GLIST. If a certification pass was not performed the GROWN DEFECTS DURING CERTIFICATION field shall be set to zero.

The Total Blocks Reallocated During Format parameter is a count of the total number of logical blocks that were reallocated during the most recent successful FORMAT UNIT command.

The Total New Blocks Reallocated parameter is a count of the total number of logical blocks that have been reallocated since the completion of the most recent successful FORMAT UNIT command.

The Power On Minutes Since Format parameter represents the unsigned number of usage minutes (i.e., minutes with power applied regardless of power state) that have elapsed since the most recent successful FORMAT UNIT command.

Upon receiving the FORMAT UNIT command, the device server should set all parameters within the Format Status log page to indicate that no such information is available. Only upon successful completion of the FORMAT UNIT command should the device server update the affected fields.

The target save disable (TSD) bit in the PARAMETER CONTROL byte (see SPC-4) shall always be set to zero to indicate that the device server provides an implicit saving frequency.

NOTE 22 - Removable media device servers may save log page information with the medium in a vendor-specific manner and location.

#### 6.2.4 Non-volatile Cache log page

The Non-volatile Cache log page (see table 113) indicates the status of battery backup for a non-volatile cache.

**Table 113 — Non-volatile Cache log page**

Byte	Bit	7	6	5	4	3	2	1	0
0		DS	SPF	PAGE CODE (17h)					
1		SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n - 3)							
3									
4		Non-volatile cache log parameters							
n									

The PAGE CODE field is defined in SPC-4 and shall be set to the value defined in table 113.

The PAGE LENGTH field is defined in SPC-4.

Table 114 defines the parameter codes.

**Table 114 — Non-volatile Cache log parameters**

Parameter code	Description
0000h	Remaining Non-volatile Time
0001h	Maximum Non-volatile Time
All others	Reserved

The Remaining Non-volatile Time parameter has the format shown in table 115.

**Table 115 — Remaining Non-volatile Time parameter data**

Bit	7	6	5	4	3	2	1	0
Byte	PARAMETER LENGTH (03h)							
0								
1	(MSB)	REMAINING NON-VOLATILE TIME						
3								(LSB)

The PARAMETER LENGTH field shall be set to the value defined in table 115.

The REMAINING NON-VOLATILE TIME field is defined in table 116.

**Table 116 — REMAINING NON-VOLATILE TIME field**

Code	Description
00_0000h	Non-volatile cache is volatile, either permanently or temporarily (e.g., if batteries need to be recharged).
00_0001h	Non-volatile cache is expected to remain non-volatile for an unknown amount of time (e.g., if battery status is unknown)
00_0002h to FF_FFFEh	Non-volatile cache is expected to remain non-volatile for the number of minutes indicated (e.g., battery-backed random access memory).
FF_FFFFh	Non-volatile cache is indefinitely non-volatile.

The Maximum Non-volatile Time parameter has the format shown in table 117.

**Table 117 — Maximum Non-volatile Time parameter data**

Bit	7	6	5	4	3	2	1	0
Byte	PARAMETER LENGTH (03h)							
0								
1	(MSB)	MAXIMUM NON-VOLATILE TIME						
3								(LSB)

The PARAMETER LENGTH field shall be set to the value defined in table 117.

The MAXIMUM NON-VOLATILE TIME field is defined in table 118.

**Table 118 — MAXIMUM NON-VOLATILE TIME field**

Code	Description
00_0000h	Non-volatile cache is volatile
00_0001h	Reserved
00_0002h to FF_FFFEh	Non-volatile cache is capable of being non-volatile for the estimated number of minutes indicated. If the time is based on batteries, it shall be based on the last full charge capacity rather than the design capacity of the batteries.
FF_FFFFh	Non-volatile cache is indefinitely non-volatile.

## 6.3 Mode parameters

### 6.3.1 Mode parameters overview

This subclause defines the block descriptors and mode pages used with direct-access block devices.

The mode parameter list, including the mode parameter header, is described in SPC-4. Direct-access block devices support zero or one mode parameter block descriptors (i.e., the block descriptor is shared by all the logical blocks on the medium).

The MEDIUM TYPE field in the mode parameter header (see SPC-4) shall be set to 00h.

The DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see SPC-4) is defined for direct-access block devices in table 119.

**Table 119 — DEVICE-SPECIFIC PARAMETER field for direct-access block devices**

Bit	7	6	5	4	3	2	1	0
	WP	Reserved		DPOFUA	Reserved			

When used with the MODE SELECT command, the write protect (WP) bit is not defined.

When used with the MODE SENSE command, a WP bit set to one indicates that the medium is write-protected. A WP bit set to zero indicates that the medium is not write-protected. When the software write protect (SWP) bit in the Control mode page (see SPC-4) is set to one, the WP bit shall be set to one. When the SWP bit in the Control mode page is set to zero, the WP bit shall be set to one if the medium is write-protected (e.g., due to mechanisms outside the scope of this standard) or zero if the medium is not write-protected.

When used with the MODE SELECT command, the DPOFUA bit is reserved.

When used with the MODE SENSE command, a DPOFUA bit set to zero indicates that the device server does not support the DPO and FUA bits. When used with the MODE SENSE command, a DPOFUA bit set to one indicates that the device server supports the DPO and FUA bits (see 4.11).

The mode page codes and subpage codes for direct-access block devices are shown in table 120.

**Table 120 — Mode page codes for direct-access block devices**

Mode page code	Description	Reference
00h	Vendor-specific (does not require page format)	
00h to 3Eh/FFh	Return all subpages <sup>a</sup>	SPC-4
01h	Read-Write Error Recovery mode page	6.3.5
02h	Disconnect-Reconnect mode page	SPC-4
03h	Obsolete (Format Device mode page)	
04h	Obsolete (Rigid Disk Geometry mode page)	
05h	Obsolete (Flexible Disk mode page)	
06h	Reserved	
07h	Verify Error Recovery mode page	6.3.6
08h	Caching mode page	6.3.4
09h	Obsolete	
0Ah/00h	Control mode page	SPC-4
0Ah/01h	Control Extension mode page	SPC-4
0Ah/02h to 3Eh	Reserved	
0Bh	Obsolete (Medium Types Supported mode page)	
0Ch	Obsolete (Notch And Partition mode page)	
0Dh	Obsolete	
0Eh to 0Fh	Reserved	
10h	XOR Control mode page	6.3.7
11h to 13h	Reserved	
14h	Enclosure Services Management mode page <sup>b</sup>	SES-2
15h to 17h	Reserved	
18h	Protocol-Specific LUN mode page	SPC-4
19h	Protocol-Specific Port mode page	SPC-4
1Ah	Power Condition mode page	SPC-4
1Bh	Reserved	
1Ch/00h	Informational Exceptions Control mode page	SPC-4
1Ch/01h	Background Control mode page	6.3.3
1Dh to 1Fh	Reserved	
20h to 3Eh	Vendor-specific (does not require page format)	
3Fh/00h	Return all mode pages <sup>a</sup>	SPC-4
3Fh/01h to 3Eh	Reserved	
3Fh/FFh	Return all mode pages and subpages <sup>a</sup>	SPC-4
<sup>a</sup> Valid only for the MODE SENSE command		
<sup>b</sup> Valid only if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-4)		

### 6.3.2 Mode parameter block descriptors

#### 6.3.2.1 Mode parameter block descriptors overview

If the device server returns a mode parameter block descriptor, it shall return a short LBA mode parameter block descriptor (see 6.3.2.2) in the mode parameter data in response to:

- a) a MODE SENSE (6) command; or
- b) a MODE SENSE (10) command with the LLBAA bit set to zero.

If the device server returns a mode parameter block descriptor and the number of logical blocks is greater than FFFF\_FFFFh, it may return a long LBA mode parameter block descriptor (see 6.3.2.3) in the mode parameter data in response to a MODE SENSE (10) command with the LLBAA bit set to one.

If the application client sends a mode parameter block descriptor in the mode parameter list, it shall send a short LBA mode parameter block descriptor (see 6.3.2.2) for a MODE SELECT (6) command.

If the application client sends a mode parameter block descriptor in the mode parameter list, it may send a long LBA mode parameter block descriptor (see 6.3.2.3) for a MODE SELECT (10) command.

Support for the mode parameter block descriptors is optional. The device server shall establish a unit attention condition with the additional sense code set to MODE PARAMETERS CHANGED (see SPC-4 and SAM-4) when the block descriptor values are changed.

#### 6.3.2.2 Short LBA mode parameter block descriptor

Table 121 defines the block descriptor for direct-access block devices used:

- a) with the MODE SELECT (6) and MODE SENSE (6) commands; and
- b) with the MODE SELECT (10) and MODE SENSE (10) commands when the LONGLBA bit is set to zero in the mode parameter header (see SPC-4).

**Table 121 — Short LBA mode parameter block descriptor**

Byte	Bit	7	6	5	4	3	2	1	0
0	(MSB)	NUMBER OF LOGICAL BLOCKS							
3									
4		Reserved							
5	(MSB)	LOGICAL BLOCK LENGTH							
7									

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.13) rather than the MODE SENSE command.

On a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the short LBA mode parameter block descriptor.

On a MODE SENSE command, if the number of logical blocks on the medium exceeds the maximum value that is able to be specified in the NUMBER OF LOGICAL BLOCKS field, the device server shall return a value of FFFF\_FFFFh.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command (see SPC-4), the value in the NUMBER OF LOGICAL BLOCKS field is ignored.

If the device supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

- a) If the NUMBER OF LOGICAL BLOCKS field is set to zero, the logical unit shall retain its current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, the logical unit shall be set to its maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
- b) If the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to its maximum capacity, the logical unit shall be set to that number of logical blocks. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, the logical unit shall not become format corrupt. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I\_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
- c) If the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFF\_FFFFh, then the MODE SELECT command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous logical block descriptor settings; or
- d) If the NUMBER OF LOGICAL BLOCKS field is set to FFFF\_FFFFh, the logical unit shall be set to its maximum capacity. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, the logical unit shall not become format corrupt. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I\_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

The LOGICAL BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.2) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current logical block length shall be returned (e.g., if the logical block length is 512 bytes and a MODE SELECT command occurs with the LOGICAL BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the LOGICAL BLOCK LENGTH field). To determine the logical block length at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.13) rather than the MODE SELECT command.

### 6.3.2.3 Long LBA mode parameter block descriptor

Table 122 defines the block descriptor for direct-access block devices used with the MODE SELECT (10) command and MODE SENSE (10) command when the LONGLBA bit is set to one in the mode parameter header (see SPC-4).

**Table 122 — Long LBA mode parameter block descriptor**

Byte	Bit	7	6	5	4	3	2	1	0
0	(MSB)	NUMBER OF LOGICAL BLOCKS							
7									
8		Reserved							
11									
12	(MSB)	LOGICAL BLOCK LENGTH							
15									

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.13) rather than the MODE SENSE command.

On a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the long LBA mode parameter block descriptor.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command (see SPC-4), the value in the NUMBER OF LOGICAL BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

- a) If the NUMBER OF LOGICAL BLOCKS field is set to zero, the logical unit shall retain its current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, the logical unit shall be set to its maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
- b) If the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to its maximum capacity, the logical unit shall be set to that number of logical blocks. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, the logical unit shall not become format corrupt. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I\_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
- c) If the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFF\_FFFF\_FFFF\_FFFFh, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous block descriptor settings; or
- d) If the NUMBER OF LOGICAL BLOCKS field is set to FFFF\_FFFF\_FFFF\_FFFFh, the logical unit shall be set to its maximum capacity. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, the logical unit shall not become format corrupt. This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I\_T nexus losses. If the content of



the LOGICAL BLOCK LENGTH field is the same as the current logical block length this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

The LOGICAL BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.2) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current logical block length shall be returned (e.g., if the logical block length is 512 bytes and a MODE SELECT command occurs with the LOGICAL BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the LOGICAL BLOCK LENGTH field). To determine the logical block length at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.13) rather than the MODE SELECT command.

### 6.3.3 Background Control mode page

The Background Control mode page (see table 123) is a subpage of the Informational Exception Control mode page (see SPC-4) and provides controls over background operations. The mode page policy (see SPC-4) for this subpage shall be shared.

**Table 123 — Background Control mode page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PS	SPF (1b)	PAGE CODE (1Ch)					
1		SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (000Ch)							
3		(LSB)							
4		Reserved					S_L_FULL	LOWIR	EN_BMS
5		Reserved							EN_PS
6	(MSB)	BACKGROUND MEDIUM SCAN INTERVAL TIME							
7		(LSB)							
8	(MSB)	BACKGROUND PRE-SCAN TIME LIMIT							
9		(LSB)							
10	(MSB)	MINIMUM IDLE TIME BEFORE BACKGROUND SCAN							
11		(LSB)							
12	(MSB)	MAXIMUM TIME TO SUSPEND BACKGROUND SCAN							
13		(LSB)							
14		Reserved							
15		Reserved							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location. If the PS is set to one in MODE SENSE data then the mode page shall be savable by issuing a MODE SELECT command with the SP bit set to one.

A SPF bit set to one indicates that the sub\_page mode page format is being used (see SPC-4).

The PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 123.

A suspend on log full (S\_L\_FULL) bit set to zero specifies that the device server shall continue running a background scan even if the Background Scan Results log page contains the maximum number of log parameters supported by the logical unit. A S\_L\_FULL bit set to one specifies that the device server shall suspend a background scan if the Background Scan Results log page contains the maximum number of log parameters supported by the logical unit.

A log only when intervention required (LOWIR) bit set to zero specifies that the device server shall log all suspected medium errors in the Background Scan Results log page (see 6.2.2). A LOWIR bit set to one specifies that the device server shall only log medium errors requiring application client intervention in the Background Scan Results log page as defined in 6.2.2.

An enable pre-scan (EN\_PS) bit set to zero specifies that background pre-scan shall be disabled. If a background pre-scan operation is in progress when EN\_PS is changed from a one to a zero, then the background pre-scan operation shall be halted before completion of the MODE SELECT command. An EN\_PS bit set to one specifies that a background pre-scan operation shall be started after the next power on. Once this background pre-scan has completed, another background pre-scan shall not occur unless the EN\_PS bit is set to zero, then set to one, and another power on occurs.

An enable background medium scan (EN\_BMS) bit set to zero specifies that background medium scan shall be disabled. An EN\_BMS bit set to one specifies that background medium scan shall be enabled. If the EN\_PS bit is also set to one then a background medium scan operation shall not start until after the background pre-scan operation is halted or completed. If a background medium scan is in progress when the EN\_BMS bit is changed from one to zero, then the background medium scan shall be suspended before completing the MODE SELECT command and remain suspended until the EN\_BMS bit is set to one, at which time the background medium scan shall resume from the suspended location.

The BACKGROUND MEDIUM SCAN INTERVAL TIME field specifies the minimum time, in hours, between the start of one background pre-scan or background medium scan operation and the start of the next background medium scan operation. If the current background medium scan operation takes longer than the value specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field, then the current background pre-scan or background medium scan continues until completion and the next background medium scan operation starts on completion of the current background pre-scan or background medium scan.

The BACKGROUND PRE-SCAN TIME LIMIT field specifies the maximum time, in hours, for a background pre-scan operation to complete. If the background pre-scan operation does not complete within the specified time then it is halted. A value of zero specifies an unlimited timeout value.

The MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field specifies the time, in milliseconds, that the logical unit shall be idle before resuming a background pre-scan or a background medium scan.

The MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field specifies the time, in milliseconds, that the logical unit should take to start processing a command received while it is performing a background pre-scan or a background medium scan.

### 6.3.4 Caching mode page

The Caching mode page (see table 124) defines the parameters that affect the use of the cache.

**Table 124 — Caching mode page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PS	SPF (0b)	PAGE CODE (08h)					
1		PAGE LENGTH (12h)							
2		IC	ABPF	CAP	DISC	SIZE	WCE	MF	RCD
3		DEMAND READ RETENTION PRIORITY				WRITE RETENTION PRIORITY			
4		(MSB)	DISABLE PRE-FETCH TRANSFER LENGTH						(LSB)
5									
6		(MSB)	MINIMUM PRE-FETCH						(LSB)
7									
8		(MSB)	MAXIMUM PRE-FETCH						(LSB)
9									
10		(MSB)	MAXIMUM PRE-FETCH CEILING						(LSB)
11									
12		FSW	LBCSS	DRA	Vendor-specific		Reserved		NV_DIS
13		NUMBER OF CACHE SEGMENTS							
14		(MSB)	CACHE SEGMENT SIZE						(LSB)
15									
16		Reserved							
17									
19		Obsolete							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location. If the PS is set to one in MODE SENSE data then the mode page shall be savable by issuing a MODE SELECT command with the SP bit set to one.

The SubPage Format (SPF) bit, PAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 124.

An initiator control (IC) enable bit set to one specifies that the device server use one of the following fields to control the caching algorithm rather than the device server's own adaptive algorithm:

- the NUMBER OF CACHE SEGMENTS field, if the SIZE bit is set to zero; or
- the CACHE SEGMENT SIZE field, if the SIZE bit is set to one.

An abort pre-fetch (ABPF) bit set to one when the DRA bit is set to zero specifies that the device server abort a pre-fetch upon receipt of a new command. An ABPF bit set to one takes precedence over the value specified in the MINIMUM PRE-FETCH field. An ABPF bit set to zero when the DRA bit set to zero specifies that the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is vendor-specific.

A caching analysis permitted (CAP) bit set to one specifies that the device server perform caching analysis during subsequent operations. A CAP bit set to zero specifies that caching analysis is disabled (e.g., to reduce overhead time or to prevent nonpertinent operations from impacting tuning values).

A discontinuity (DISC) bit set to one specifies that the device server continue the pre-fetch across time discontinuities (e.g., across cylinders) up to the limits of the buffer, or segment, space available for the pre-fetch. A DISC bit set to zero specifies that pre-fetches be truncated or wrapped at time discontinuities.

A size enable (SIZE) bit set to one specifies that the CACHE SEGMENT SIZE field be used to control caching segmentation. A SIZE bit set to zero specifies that the NUMBER OF CACHE SEGMENTS field be used to control caching segmentation. Simultaneous use of both the number of segments and the segment size is vendor-specific.

A writeback cache enable (WCE) bit set to zero specifies that the device server shall complete a WRITE command with GOOD status only after writing all of the data to the medium without error. A WCE bit set to one specifies that the device server may complete a WRITE command with GOOD status after receiving the data without error and prior to having written the data to the medium.

A multiplication factor (MF) bit set to zero specifies that the device server shall interpret the MINIMUM PRE-FETCH field and the MAXIMUM PRE-FETCH field in terms of the number of logical blocks for each of the respective types of pre-fetch. An MF bit set to one specifies that the device server shall interpret the MINIMUM PRE-FETCH field and the MAXIMUM PRE-FETCH field to be specified in terms of a scalar number that, when multiplied by the number of logical blocks to be transferred for the current command, yields the number of logical blocks for each of the respective types of pre-fetch.

A read cache disable (RCD) bit set to zero specifies that the device server may return data requested by a READ command by accessing either the cache or medium. A RCD bit set to one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache).

The DEMAND READ RETENTION PRIORITY field (see table 125) specifies the retention priority the device server should assign for data read into the cache that has also been transferred to the data-in buffer.

**Table 125 — DEMAND READ RETENTION PRIORITY field**

Code	Description
0h	The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch).
1h	The device server should replace data put into the cache via a READ command sooner (i.e., read data has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h to Eh	Reserved
Fh	The device server should not replace data put into the cache via a READ command if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and the data in the cache may be replaced.

The WRITE RETENTION PRIORITY field (see table 126) specifies the retention priority the device server should assign for data written into the cache that has also been transferred from the cache to the medium.

**Table 126 — WRITE RETENTION PRIORITY field**

Code	Description
0h	The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch).
1h	The device server should replace data put into the cache during a WRITE command or a WRITE AND VERIFY command sooner (i.e., has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h to Eh	Reserved
Fh	The device server should not replace data put into the cache during a WRITE command or a WRITE AND VERIFY command if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and the data in the cache may be replaced.

An anticipatory pre-fetch occurs when data is placed in the cache that has not been requested. This may happen in conjunction with the reading of data that has been requested. The DISABLE PRE-FETCH TRANSFER LENGTH field, the MINIMUM PRE-FETCH field, the MAXIMUM PRE-FETCH field, and the MAXIMUM PRE-FETCH CEILING field give an indication to the device server how it should manage the cache based on the most recent READ command. An anticipatory pre-fetch may occur based on other information. These fields are only recommendations to the device server and should not cause a CHECK CONDITION to occur if the device server is not able to satisfy the request.

The DISABLE PRE-FETCH TRANSFER LENGTH field specifies the selective disabling of anticipatory pre-fetch on long transfer lengths. The value in this field is compared to the transfer length requested by a READ command. If the transfer length is greater than the disable pre-fetch transfer length, then an anticipatory pre-fetch is not done for the command. Otherwise the device server should attempt an anticipatory pre-fetch. If the DISABLE PRE-FETCH TRANSFER LENGTH field is set to zero, then all anticipatory pre-fetching is disabled for any request for data, including those with a transfer length of zero.

The MINIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch regardless of the delays it might cause in processing subsequent commands. The field contains either:

- a) a number of logical blocks, if the MF bit is set to zero; or
- b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The pre-fetching operation begins at the logical block after the last logical block of a READ command. Pre-fetching shall always halt when it reaches the last logical block on the medium. Errors that occur during the pre-fetching operation shall not be reported to the application client unless the device server is unable to process subsequent commands correctly as a result of the error. In this case the error may be reported either as an error for that subsequent command, or as a deferred error, at the discretion of the device server and according to the rules for reporting deferred errors (see SPC-4).

If the pre-fetch has read more than the amount of data specified by the MINIMUM PRE-FETCH field, then pre-fetching should be terminated whenever another command enters the enabled state (see SAM-4). This requirement is ignored when the MINIMUM PRE-FETCH field value is equal to the MAXIMUM PRE-FETCH field value.

The MAXIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch if the pre-fetch does not delay processing of subsequent commands. The field contains either:

- a) a number of logical blocks, if the MF bit is set to zero; or
- b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre-fetch as a result of one READ command. It is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH field and MAXIMUM PRE-FETCH CEILING field to trade off pre-fetching new data with displacing old data already stored in the cache.

The MAXIMUM PRE-FETCH CEILING field specifies an upper limit on the number of logical blocks computed as the maximum pre-fetch. If this number of logical blocks is greater than the value in the MAXIMUM PRE-FETCH field, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

NOTE 23 - If the MF bit is set to one the MAXIMUM PRE-FETCH CEILING field is useful in limiting the amount of data to be pre-fetched.

A force sequential write (FSW) bit set to one specifies that, for commands writing to more than one logical block, the device server shall write the logical blocks to the medium in ascending sequential order. An FSW bit set to zero specifies that the device server may reorder the sequence of writing logical blocks (e.g., in order to achieve faster command completion).

A logical block cache segment size (LBCSS) bit set to one specifies that the CACHE SEGMENT SIZE field units shall be interpreted as logical blocks. An LBCSS bit set to zero specifies that the CACHE SEGMENT SIZE field units shall be interpreted as bytes. The LBCSS shall not impact the units of other fields.

A disable read-ahead (DRA) bit set to one specifies that the device server shall not read into the pre-fetch buffer any logical blocks beyond the addressed logical block(s). A DRA bit set to zero specifies that the device server may continue to read logical blocks into the pre-fetch buffer beyond the addressed logical block(s).

The NUMBER OF CACHE SEGMENTS field specifies the number of segments into which the device server shall divide the cache.

The CACHE SEGMENT SIZE field specifies the segment size in bytes if the LBCSS bit is set to zero or in logical blocks if the LBCSS bit is set to one. The CACHE SEGMENT SIZE field is valid only when the SIZE bit is set to one.

An NV\_DIS bit set to one specifies that the device server shall disable a non-volatile cache and indicates that a non-volatile cache is supported but disabled. An NV\_DIS bit set to zero specifies that the device server may use a non-volatile cache and indicates that a non-volatile cache may be present and enabled.

### 6.3.5 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 127) specifies the error recovery parameters the device server shall use during any command that performs a read or write operation to the medium (e.g., READ command, WRITE command, or a WRITE AND VERIFY command).

**Table 127 — Read-Write Error Recovery mode page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PS	SPF (0b)	PAGE CODE (01h)					
1		PAGE LENGTH (0Ah)							
2		AWRE	ARRE	TB	RC	Error recovery bits			
						EER	PER	DTE	DCR
3		READ RETRY COUNT							
4		Obsolete							
5		Obsolete							
6		Obsolete							
7		Reserved						Restricted for MMC-6	
8		WRITE RETRY COUNT							
9		Reserved							
10		(MSB)							
11		RECOVERY TIME LIMIT							
		(LSB)							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

The SubPage Format (SPF) bit, PAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 127.

An automatic write reallocation enabled (AWRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during write operations.

An AWRE bit set to one specifies that the device server shall enable automatic reallocation of defective logical blocks during write operations. The automatic reallocation shall be performed only if the device server has the valid data (e.g., original data in a buffer or recovered from the medium). The valid data shall be placed in the reallocated logical block. The device server shall report any failures that occur during the reallocation operation. Error reporting as specified by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be performed only after completion of the reallocation. See the REASSIGN BLOCKS command (see 5.18) for error procedures.

An automatic read reallocation enabled (ARRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during read operations.

An ARRE bit set to one specifies that the device server shall enable automatic reallocation of defective logical blocks during read operations. All error recovery actions required by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be processed. The automatic reallocation shall then be performed only if the device server successfully recovers the data. The recovered data shall be placed in the reallocated logical block. The device server shall report any failures that occur during the reallocation operation. Error reporting as specified by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be performed only after completion of the reallocation operation. See the REASSIGN BLOCKS command (see 5.18) for error procedures.

A transfer block (TB) bit set to zero specifies that if an unrecovered read error occurs during a read operation, then the device server shall not transfer any data for the logical block to the data-in buffer. A TB bit set to one specifies that if an unrecovered read error occurs during a read operation, then the device server shall transfer pseudo read data before returning CHECK CONDITION status. The data returned in this case is vendor-specific. The TB bit does not affect the action taken for recovered read errors.

A read continuous (RC) bit set to zero specifies that error recovery operations that cause delays during the data transfer are acceptable. Data shall not be fabricated.

An RC bit set to one specifies the device server shall transfer the entire requested length of data without adding delays during the data transfer to perform error recovery procedures. The device server may transfer pseudo read in order to maintain a continuous flow of data. The device server shall assign priority to the RC bit over conflicting bits within this byte.

NOTE 24 - The RC bit may be set to one in image processing, audio, or video applications.

An enable early recovery (EER) bit set to one specifies that the device server shall use the most expedient form of error recovery first. An EER bit set to zero specifies that the device server shall use an error recovery procedure that minimizes the risk of error mis-detection or mis-correction. This bit only applies to data error recovery and it does not affect positioning retries.

NOTE 25 - An EER bit set to one may imply an increase in the probability of error mis-detection or mis-correction. An EER bit set to zero allows the specified retry limit to be exhausted prior to using additional information (e.g., ECC bytes) to correct the error.

A post error (PER) bit set to one specifies that if a recovered read error occurs during a command performing a read or write operation, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. A PER bit set to zero specifies that if a recovered read error occurs during a command performing a read or write operation, then the device server shall terminate the command with CHECK CONDITION status, and shall perform error recovery procedures within the limits established by the error recovery parameters. If the DTE bit is set to one, then the PER bit shall be set to one.

A data terminate on error (DTE) bit set to one specifies that the device server shall terminate the data-in or data-out buffer transfer of a command performing a read or write operation upon detection of a recovered error. A DTE bit set to zero specifies that the device server shall not terminate the data-in or data-out buffer transfer of a command performing a read or write operation upon detection of a recovered error.

A disable correction (DCR) bit set to one specifies that additional information (e.g., ECC bytes) (see 4.4) shall not be used for data error recovery. A DCR bit set to zero allows the use of additional information (e.g., ECC bytes) for data error recovery. If the EER bit is set to one, the DCR bit shall be set to zero.



The combinations of the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) are explained in table 128.

**Table 128 — Combined error recovery bit descriptions (part 1 of 4)**

EER	PER	DTE	DCR	Description
0	0	0	0	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations and shall perform error correction in an attempt to recover the data.</p> <p>The device server shall not report recovered read errors. The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
0	0	0	1	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations but shall not perform error correction in an attempt to recover the data.</p> <p>The device server shall not report recovered read errors. The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
0	0	1	0	Invalid mode. The PER bit shall be set to one if the DTE bit is set to one. <sup>a</sup>
0	0	1	1	
<sup>a</sup> If an invalid combination of the error recovery bits is sent by the application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.				

**Table 128 — Combined error recovery bit descriptions (part 2 of 4)**

EER	PER	DTE	DCR	Description
0	1	0	0	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations and shall perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p> <p>If a recovered error occurs while the device server is performing a read or write operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command.</p>
0	1	0	1	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations but shall not perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p> <p>If a recovered error occurs while the device server is performing a read or write operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command.</p>
<sup>a</sup> If an invalid combination of the error recovery bits is sent by the application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.				

**Table 128 — Combined error recovery bit descriptions** (part 3 of 4)

EER	PER	DTE	DCR	Description
0	1	1	0	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations and shall perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of error.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
0	1	1	1	<p>The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read operations, the WRITE RETRY COUNT field for write operations, and the VERIFY RETRY COUNT field (see 6.3.6) for verify operations but shall not perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of the error.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
1	0	0	0	<p>The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.</p> <p>The device server shall not report recovered errors. The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
1	0	0	1	Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. <sup>a</sup>
1	0	1	0	Invalid mode. The PER bit shall be set to one if the DTE bit is set to one. <sup>a</sup>
1	0	1	1	
<sup>a</sup> If an invalid combination of the error recovery bits is sent by the application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.				

**Table 128 — Combined error recovery bit descriptions (part 4 of 4)**

EER	PER	DTE	DCR	Description
1	1	0	0	<p>The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p> <p>If a recovered error occurs while the device server is performing a read or write operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command.</p>
1	1	0	1	Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. <sup>a</sup>
1	1	1	0	<p>The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.</p> <p>The device server shall terminate a command performing a read or write operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of the error.</p> <p>If an unrecovered read error occurs during a read operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the data-in buffer.</p>
1	1	1	1	Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. <sup>a</sup>
<sup>a</sup> If an invalid combination of the error recovery bits is sent by the application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.				

The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read operations.

The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write operations.

The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. The device server may round this value as described in SPC-4. The limit in this field specifies the maximum error recovery time allowed for any individual logical block. A RECOVERY TIME LIMIT field set to zero specifies that the device server shall use its default value.

When both a retry count and a recovery time limit are specified, the field that specifies the recovery action of least duration shall have priority.

NOTE 26 - To disable all types of correction and retries the application client should set the EER bit to zero, the PER bit to one, the DTE bit to one, the DCR bit to one, the READ RETRY COUNT field to 00h, the WRITE RETRY COUNT field to 00h, and the RECOVERY TIME LIMIT field to 0000h.

### 6.3.6 Verify Error Recovery mode page

The Verify Error Recovery mode page (see table 129) specifies the error recovery parameters the device server shall use during the VERIFY command and the verify operation of the WRITE AND VERIFY command.

**Table 129 — Verify Error Recovery mode page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PS	SPF (0b)	PAGE CODE (07h)					
1		PAGE LENGTH (0Ah)							
2		Reserved				Error recovery bits			
						EER	PER	DTE	DCR
3		VERIFY RETRY COUNT							
4		Obsolete							
5		Reserved							
9									
10		(MSB)	VERIFY RECOVERY TIME LIMIT						
11									(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

The SubPage Format (SPF) bit, PAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 129.

The AWRE bit as defined in the Read-Write Error Recovery mode page (see 6.3.5) applies to the WRITE AND VERIFY command. The VERIFY command shall not perform automatic reallocation.

The EER bit, the PER bit, the DTE bit, and the DCR bit (i.e., the error recovery bits) are defined in 6.3.5. The combinations of these bits are defined in table 128 (see 6.3.5).

The VERIFY RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during a verify operation.

The VERIFY RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use error recovery procedures to recover data for an individual logical block. The device server may round this value as described in SPC-4.

When both a retry count and a recovery time limit are specified, the one that requires the least time for data error recovery actions shall have priority.

**NOTE 27** - To disable all types of correction and retries the application client should set the EER bit to zero, the PER bit to one, the DTE bit to one, the DCR bit to one, the VERIFY RETRY COUNT field to 00h, and the VERIFY RECOVERY TIME LIMIT field to 0000h.

### 6.3.7 XOR Control mode page

The XOR Control mode page (see table 130) provides the application client with the means to obtain or modify certain XOR operating parameters of the logical unit.

**Table 130 — XOR Control mode page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PS	SPF (0b)	PAGE CODE (10h)					
1		PAGE LENGTH (16h)							
2		Reserved						XORDIS	Reserved
3		Reserved							
4	(MSB)	MAXIMUM XOR WRITE SIZE							
7		(LSB)							
8		Reserved							
11		Reserved							
12		Obsolete							
19		Reserved							
20		Reserved							
21		Reserved							
22		Obsolete							
23		Obsolete							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

The SubPage Format (SPF) bit, PAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 130.

An XOR disable (XORDIS) bit set to zero specifies that the device server shall enable processing of XOR commands (i.e., the XDREAD commands (see 5.40 and 5.41), the XDWRITE commands (see 5.42 and 5.43), the XDWRITEREAD commands (see 5.44 and 5.45), and the XPWRITE commands (see 5.46 and 5.47)). An XORDIS bit set to one shall disable processing of XOR commands. If the XORDIS bit is set to one and an XOR command is received, then the device server shall terminate the XOR command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.

The MAXIMUM XOR WRITE SIZE field specifies the maximum transfer length in blocks that the device server accepts for a single XDWRITE command or XDREAD command. Requests for transfer lengths exceeding this limit result in CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. The MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2) indicates the maximum value of the MAXIMUM XOR WRITE SIZE field supported by the device server.

## 6.4 Vital product data (VPD) parameters

### 6.4.1 VPD parameters overview

This subclause defines the VPD pages used only with direct-access block devices. See SPC-4 for VPD pages used with all device types.

The VPD page codes specific to direct-access block devices are defined in table 131.

**Table 131 — Direct-access block device VPD page codes**

VPD page code	VPD page name	Reference	Support requirements
B0h	Block Limits VPD page	6.4.2	Optional
B1h	Block Device Characteristics VPD page	6.4.3	Optional
B2h to BFh	Reserved for this standard		

### 6.4.2 Block Limits VPD page

The Block Limits VPD page (see table 132) provides the application client with the means to obtain certain operating parameters of the logical unit.

**Table 132 — Block Limits VPD page**

Bit	7	6	5	4	3	2	1	0
Byte								
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B0h)							
2	Reserved							
3	PAGE LENGTH (10h)							
4	Reserved							
5								
6	(MSB)	OPTIMAL TRANSFER LENGTH GRANULARITY						(LSB)
7								
8	(MSB)	MAXIMUM TRANSFER LENGTH						(LSB)
11								
12	(MSB)	OPTIMAL TRANSFER LENGTH						(LSB)
15								
16	(MSB)	MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH						(LSB)
19								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 132.

The OPTIMAL TRANSFER LENGTH GRANULARITY field indicates the optimal transfer length granularity in blocks for a single ORWRITE command, PRE-FETCH command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDREAD command, XDWRITE command, XDWRITEREAD

command, or XPWRITE command. Transfers with transfer lengths not equal to a multiple of this value may incur significant delays in processing.

The MAXIMUM TRANSFER LENGTH field indicates the maximum transfer length in blocks that the device server accepts for a single ORWRITE command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDWRITEREAD command, or XPWRITE command. Requests for transfer lengths exceeding this limit result in CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. A MAXIMUM TRANSFER LENGTH field set to zero indicates that there is no reported limit on the transfer length.

The OPTIMAL TRANSFER LENGTH field indicates the optimal transfer length in blocks for a single ORWRITE command, PRE-FETCH command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDREAD command, XDWRITE command, XDWRITEREAD command, or XPWRITE command. Transfers with transfer lengths exceeding this value may incur significant delays in processing.

The MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH field indicates:

- the maximum transfer length in blocks that the device server accepts for a single PRE-FETCH command;
- if the XOR Control mode page (see 6.3.7) is implemented, then the maximum value supported by the MAXIMUM XOR WRITE SIZE field in the XOR Control mode page; and
- if the XOR Control mode page is not implemented, then the maximum transfer length in blocks that the device server accepts for a single XDWRITE command or XDREAD command.

The device server should set the MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH field to less than or equal to the MAXIMUM TRANSFER LENGTH field.

#### 6.4.3 Block Device Characteristics VPD page

The Block Device Characteristics VPD page contains parameters indicating characteristics of the logical unit.

Table 133 defines the Block Device Characteristics VPD page.

**Table 133 — Block Device Characteristics VPD page**

Byte	Bit	7	6	5	4	3	2	1	0
0		PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1		PAGE CODE (B1h)							
2		Reserved							
3		PAGE LENGTH (3Ch)							
4		MEDIUM ROTATION RATE							
5									
6		Reserved							
7		Reserved				NOMINAL FORM FACTOR			
8		Reserved							
63									

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 133.



The MEDIUM ROTATION RATE field is defined in table 134.

**Table 134 — MEDIUM ROTATION RATE field**

Code	Description
0000h	Medium rotation rate is not reported
0001h	Non-rotating medium (e.g., solid state)
0002h to 0400h	Reserved
0401h to FFFEh	Nominal medium rotation rate in rotations per minute (i.e., rpm) (e.g., 7 200 rpm = 1C20h, 10 000 rpm = 2710h, and 15 000 rpm = 3A98h)
FFFFh	Reserved

The NOMINAL FORM FACTOR field indicates the nominal form factor of the device containing the logical unit and is defined in table 135.

**Table 135 — NOMINAL FORM FACTOR field**

Code	Description
0h	Nominal form factor is not reported
1h	5.25 inch
2h	3.5 inch
3h	2.5 inch
4h	1.8 inch
5h	Less than 1.8 inch
All others	Reserved

## Annex A

(informative)

### Numeric order codes

#### A.1 Variable length CDBs

Some commands in table 13 (see 5.1) use the variable length command format defined in SPC-4. These commands use operation code 7Fh and are differentiated by service action codes as described in table A.1.

**Table A.1 — Variable length command service action code assignments**

Operation code/service action code	Description
7Fh/0000h	Reserved
7Fh/0001h	Reserved
7Fh/0002h	Reserved
7Fh/0003h	XDREAD (32)
7Fh/0004h	XDWRITE (32)
7Fh/0005h	Reserved
7Fh/0006h	XPWRITE (32)
7Fh/0007h	XDWRITEREAD (32)
7Fh/0008h	Reserved
7Fh/0009h	READ (32)
7Fh/000Ah	VERIFY (32)
7Fh/000Bh	WRITE (32)
7Fh/000Ch	WRITE AND VERIFY (32)
7Fh/000Dh	WRITE SAME (32)
7Fh/000Eh to 07FFh	Reserved
7Fh/0800h to FFFFh	See SPC-4

#### A.2 Service action CDBs

Some commands in table 13 (see 5.1) are implemented as device-type specific service actions for the SERVICE ACTION IN (16) operation code 9Eh (see SPC-4). These commands are differentiated by service action codes as described in table A.2.

**Table A.2 — SERVICE ACTION IN (16) service actions**

Operation code/service action code	Description
9Eh/00h to 0Fh	Reserved for commands applicable to all device types (see SPC-4)
9Eh/10h	READ CAPACITY (16)
9Eh/11h	READ LONG (16)
9Eh/12h to 1Fh	Reserved

Some commands in table 13 (see 5.1) are implemented as device-type specific service actions for the SERVICE ACTION OUT (16) operation code 9Fh (see SPC-4). These commands are differentiated by service action codes as described in table A.3.

**Table A.3 — SERVICE ACTION OUT (16) service actions**

<b>Operation code/service action code</b>	<b>Description</b>
9Fh/00h to 0Fh	Reserved for commands applicable to all device types (see SPC-4)
9Fh/10h	Reserved
9Fh/11h	WRITE LONG (16)
9Fh/12h to 1Fh	Reserved

## **Annex B**

(informative)

### **XOR command examples**

#### **B.1 XOR command examples overview**

This annex provides XOR command examples in storage array controller supervised configurations.

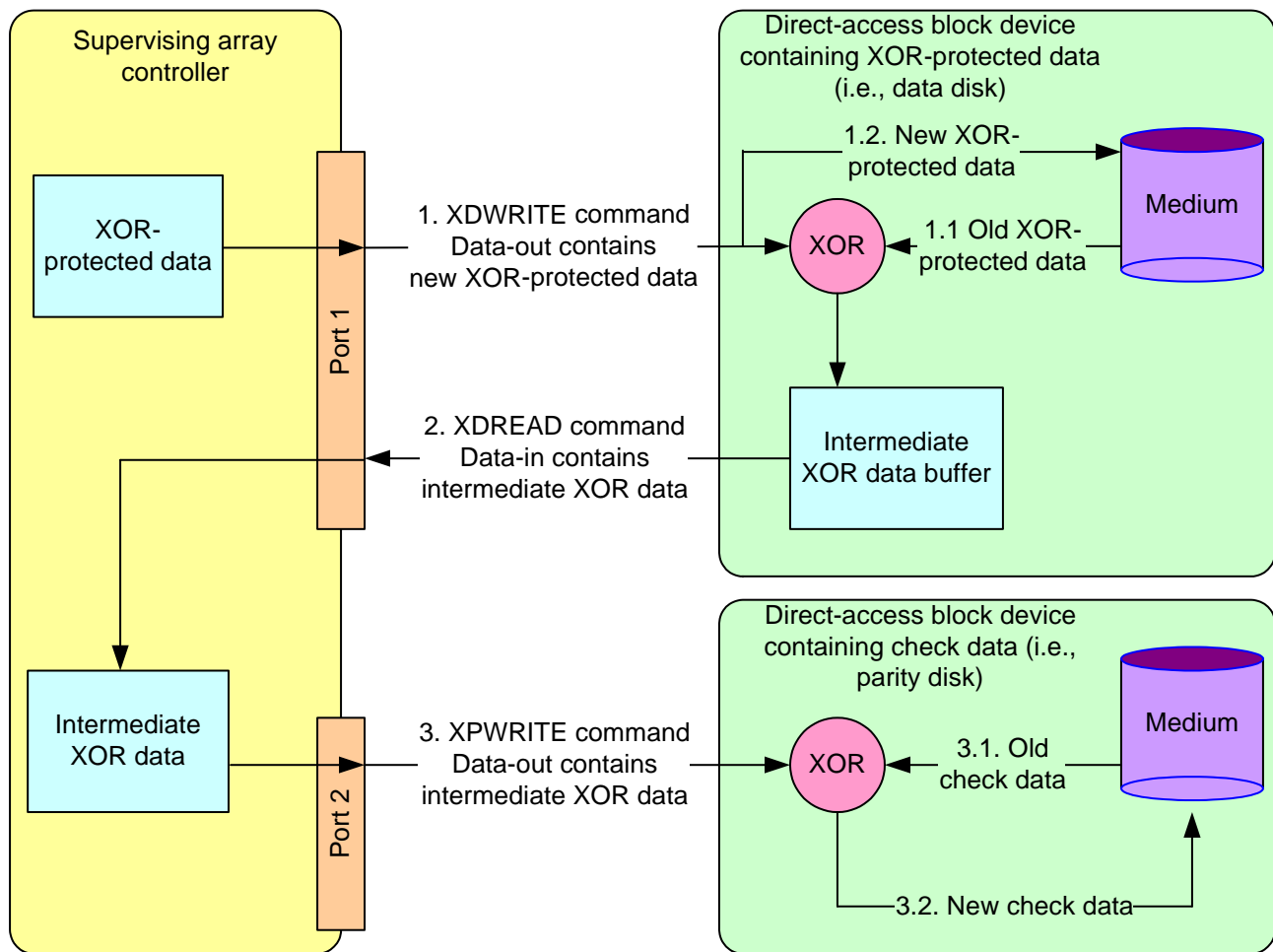
#### **B.2 Update write operation**

Figure B.1 shows a read-modify-write operation supervised by a storage array controller. The example uses a supervising storage array controller, a direct-access block device holding XOR-protected data (i.e., the data disk), and a direct-access block device holding check data (i.e., the parity disk). Three SCSI commands are used: an XDWRITE command, an XDREAD command, and an XPWRITE command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by sending XOR-protected data to the data disk using an XDWRITE command.

The data disk reads old XOR-protected data from its medium, performs an XOR operation using the old XOR-protected data and the XOR-protected data from the supervising storage array controller, stores the resulting intermediate XOR data in its buffer memory, and writes the XOR-protected data from the supervising storage array controller to its medium. The supervising storage array controller reads the resulting intermediate XOR data from the buffer memory by sending the data disk an XDREAD command.

The supervising storage array controller makes the resulting intermediate XOR data (i.e., data read with the XDREAD command) available to the parity disk by sending an XPWRITE command. The parity disk performs an XOR operation using the intermediate XOR data and the XOR data in its buffer memory. The resulting new XOR data is written to the medium.

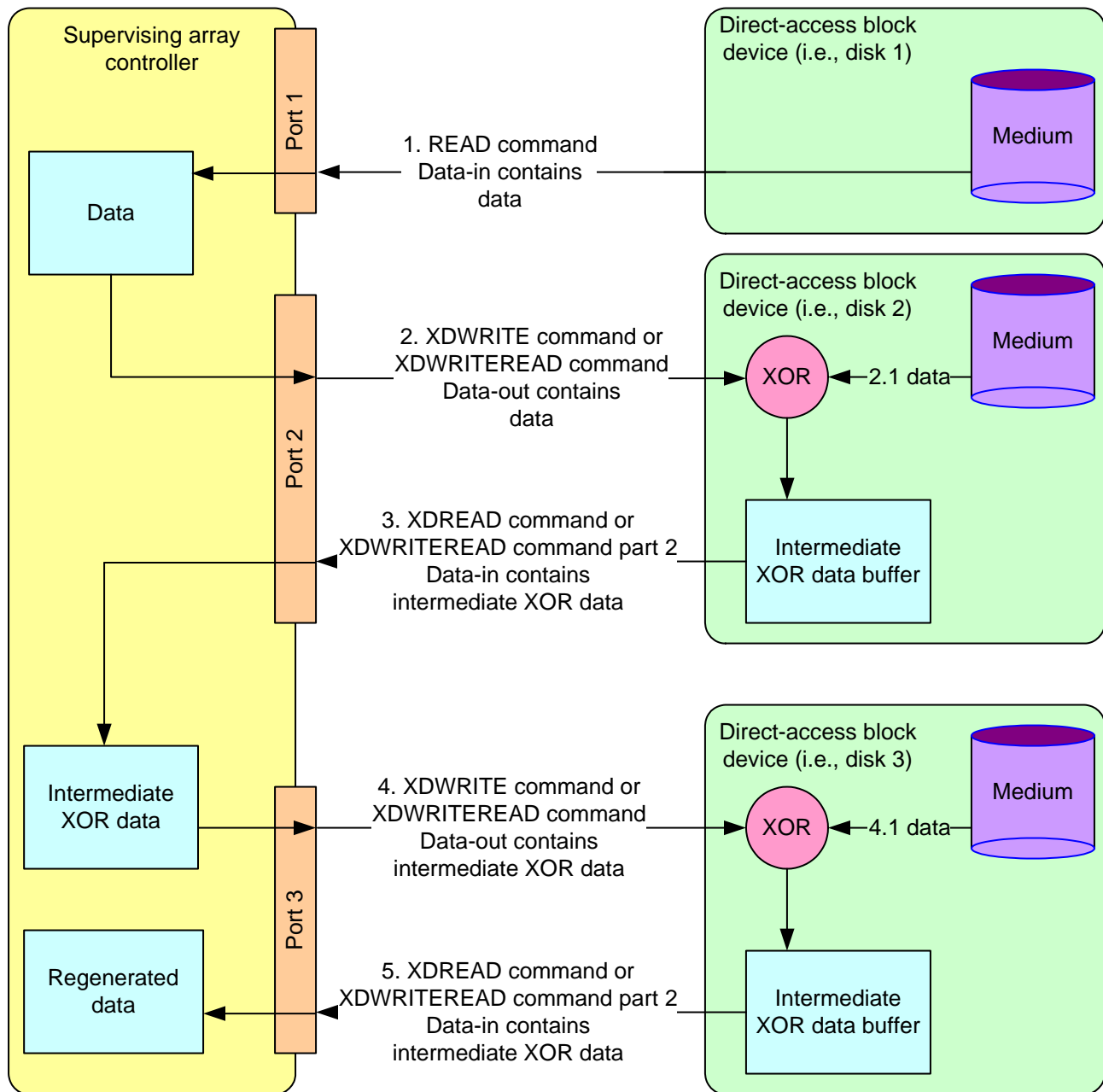


**Figure B.1 — Update write operation (storage array controller supervised)**

### B.3 Regenerate operation

Figure B.2 shows a regenerate operation supervised by a storage array controller. The example uses a supervising storage array controller and three direct-access block devices (i.e., disk 1, disk 2, and disk 3). Three SCSI commands are used: a READ command, an XDWRITE command, and an XDREAD command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by issuing a READ command to disk 1. The data received from this command is sent by the supervising storage array controller to disk 2 using an XDWRITE command with the DISABLE WRITE bit set to one. Disk 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and stores the resulting intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data from the buffer memory by issuing an XDREAD command to disk 2. The supervising storage array controller issues XDWRITE commands and XDREAD commands in the same manner to disk 3. The resulting data from disk 3 is the regenerated data.



**Figure B.2 — Regenerate operation (storage array controller supervised)**

## B.4 Rebuild operation

Figure B.3 shows a rebuild operation supervised by a storage array controller. The example uses a supervising storage array controller, two direct-access block devices (i.e., disk 1 and disk 2) as the source devices, and one direct-access block device (i.e., disk 3) as the rebuild device. Four SCSI commands are used: a READ command, an XDWRITE command, an XDREAD command, and a WRITE command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by issuing a READ command to disk 1. The data received from the READ command is sent by the supervising storage array controller to disk 2 using an XDWRITE command with a DISABLE WRITE bit set to one. Disk 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and stores the resulting

intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data by sending an XDREAD command to disk 2.

The resulting data from disk 2 is the rebuilt data and is sent to the direct-access block device being rebuilt (i.e., disk 3) using a WRITE command.

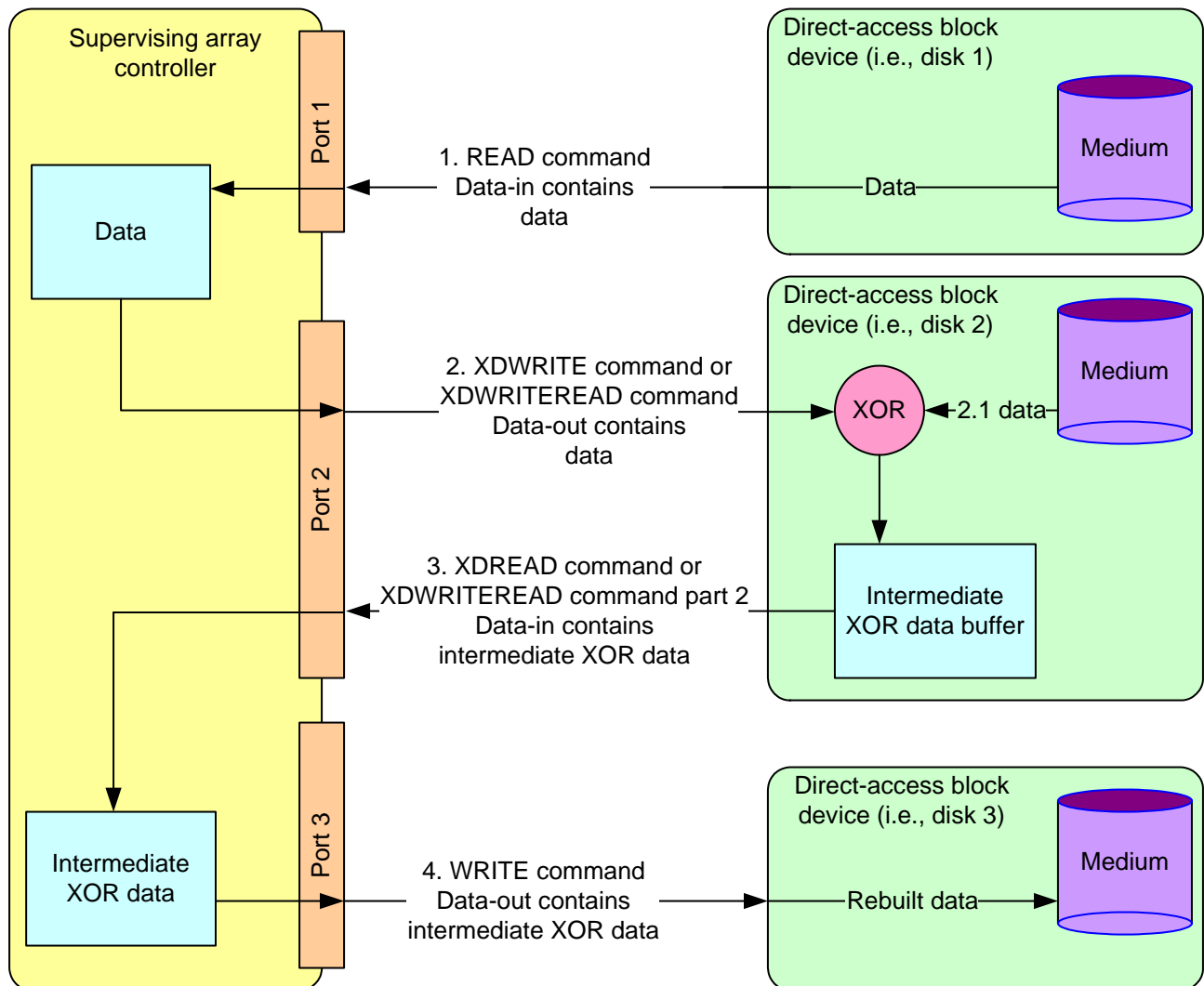


Figure B.3 — Rebuild operation (storage array controller supervised)

## Annex C

(informative)

### CRC example in C

The following is an example C program that generates the value for the LOGICAL BLOCK GUARD field in protection information (see 4.17).

```
// picrc.cpp : SCSI SBC-2 Protection Information CRC generator
#include "stdafx.h"
#include <stdio.h>
#include <malloc.h>

/* return crc value */
unsigned short calculate_crc(unsigned char *frame, unsigned long length) {
    unsigned short const poly = 0x8BB7L; /* Polynomial */
    unsigned const int poly_length = 16;
    unsigned short crc_gen;
    unsigned short x;
    unsigned int i, j, fb;
    unsigned const int invert = 0; /* 1=seed with 1s and invert the CRC */

    crc_gen = 0x0000;
    crc_gen ^= invert? 0xFFFF: 0x0000; /* seed generator */

    for (i = 0; i < length; i += 2) {
        /* assume little endian */
        x = (frame[i] << 8) | frame[i+1];

        /* serial shift register implementation */
        for (j = 0; j < poly_length; j++) {
            fb = ((x & 0x8000L) == 0x8000L) ^ ((crc_gen & 0x8000L) ==
0x8000L);
            x <<= 1;
            crc_gen <<= 1;
            if (fb)
                crc_gen ^= poly;
        }
    }

    return crc_gen ^ (invert? 0xFFFF: 0x0000); /* invert output */
} /* calculate_crc */

/* function prototype */
unsigned short calculate_crc(unsigned char *, unsigned long);

void main (void) {
    unsigned char *buffer;
    unsigned long buffer_size = 32;
    unsigned short crc;
    unsigned int i;

    /* 32 0x00 */
    buffer = (unsigned char *) malloc (buffer_size);
    for (i = 0; i < buffer_size; i++) {
        buffer[i] = 0x00;
    }
}
```



```

}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC all-zeros is %04x\n", crc);
free (buffer);

/* 32 0xFF */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
    buffer[i] = 0xFF;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC all-ones is %04x\n", crc);
free (buffer);

/* 0x00 incrementing to 0x1F */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
    buffer[i] = i;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC incrementing is %04x\n", crc);
free (buffer);

/* 0xFF 0xFF then 30 zeros */
buffer = (unsigned char *) malloc (buffer_size);
buffer[0] = 0xff;
buffer[1] = 0xff;
for (i = 2; i < buffer_size; i++) {
    buffer[i] = 0x00;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC FF FF then 30 zeros is %04x\n", crc);
free (buffer);

/* 0xFF decrementing to 0xE0 */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
    buffer[i] = 0xff - i;
}
crc = calculate_crc(buffer, buffer_size);
printf ("Example CRC FF decrementing to E0 is %04x\n", crc);
free (buffer);

} /* main */

```